



*Building the National Virtual Collaboratory
for Earthquake Engineering Research*

NEESgrid

Technical Report NEESgrid-2004-40

www.neesgrid.org

(Whitepaper Version: 1.0)

Last modified: 9/16/2004

Reference NEESgrid Data Model

Jun Peng and Kincho H. Law

Department of Civil and Environmental Engineering¹

¹ Stanford University, Stanford, CA 94305

Feedback on this document should be directed to junpeng@stanford.edu

Acknowledgment: This work was supported primarily by the George E. Brown, Jr. Network for Earthquake Engineering Simulation (NEES) Program of the National Science Foundation under Award Number CMS-0117853.

1	Introduction	4
1.1	Purpose.....	4
1.2	Definition	5
1.3	Scope.....	5
2	Data Modeling Tool and Approach.....	5
2.1	Protégé – Data Modeling Tool.....	5
2.2	Object-Oriented Data Modeling.....	6
3	Relevant Data Models	7
3.1	Oregon State Model	7
3.2	Ontology of Science.....	8
3.3	Berkeley CUREE/ Kajima	9
3.4	SensorML.....	10
3.5	Specimen Models.....	10
4	Overview of the Reference Data Model.....	11
4.1	SiteInformation	13
4.1.1	Organization	13
4.1.2	Person	13
4.1.3	Site.....	14
4.2	Activity	15
4.2.1	Project.....	15
4.2.2	Task	16
4.2.3	EventGroup.....	17
4.2.4	Event.....	17
4.2.5	MultiSiteActivity	17
4.2.6	Layout of Activity Classes	18
4.3	Apparatus	19
4.3.1	PrimaryEquipment.....	19
4.3.2	SecondaryEquipment.....	20
4.3.3	TertiaryEquipment.....	22
4.3.4	Specimen	23
4.4	ApparatusSetup.....	23
4.4.1	PhysicalSetup	24
4.4.2	DAQSetup	24
4.4.3	InputDataSetup	25
4.5	DataElement.....	25

4.5.1	Publication.....	26
4.5.2	File.....	26
4.5.3	InputData.....	27
4.6	ComplexDataType.....	27
4.6.1	Folder.....	27
4.6.2	RolePerson.....	28
4.6.3	Unit.....	28
4.6.4	Measurement.....	28
4.6.5	DateTime.....	28
4.6.6	Angle.....	29
4.6.7	Location.....	29
4.6.8	ApparatusLocation.....	30
5	Validation and Usability Test.....	30
5.1	Mini-MOST Experiment.....	30
5.2	Inputting Experimental Data.....	30
5.3	Browsing Experimental Data.....	31
6	Summary and Discussions.....	38
	Acknowledgements.....	38
	References.....	39

1 Introduction

The primary goal of the NEESgrid data/metadata effort is to work collaboratively with the NEESgrid System Integrator team and the NEES community to help define data requirements and needs for the George Brown Jr. Network for Earthquake Engineering Simulation, instigated by the National Science Foundation. The NEESgrid promotes NEES as a distributed collaborative laboratory for earthquake engineering research and simulation. The “collaboratory” will allow researchers gain remote, shared access to experimental equipment and data.

Reference data models have been developed for supporting the major activities involved in earthquake engineering experiments and simulations. The current data modeling efforts include the development of reference data models for shake table experiments [26], centrifuge experiments [30], and computer simulations [8]. The development of NEESgrid data models is based on the experience gained from the review of the state of practice in data representations, data modeling tools, relevant data models, and the suggestions/feedback from the NEESgrid team and the NEES community [23]. Sample data sets have been utilized to demonstrate the features of the data models, along with scenarios for their use. Preliminary validation and usability tests have been performed on the developed reference data model [24]. The usability test has demonstrated that the data model is sufficiently comprehensive to save and organize experimental data, such as data from Mini-MOST [21] experiments.

1.1 Purpose

In order to facilitate collaboration within the NEES framework, one of the key services that NEESgrid needs to support is with respect to the data and metadata for earthquake engineering simulations. It is well known that engineering design and manufacturing activities generally involve a large set of independent but interrelated data items [14, 15]. Traditional hierarchical, network, and relational database models, which are designed for highly structured commercial applications, do not adequately support technical engineering problem domains. To support the expressive concepts and the semantic content of engineering data, object oriented data models are often employed. Influenced by the field of artificial intelligence, semantic relationships such as classification, association, aggregation and generalization can be used for organizing and structuring engineering data [16]. Besides the mechanism needed to represent and manipulate data, data model development requires some knowledge on the intended use of the data [17]. Earthquake engineering experiments also require semantically rich data models to facilitate storage and retrieval of experimental data.

Following the NEESgrid Data and Metadata Advisory Group meeting that was held at the Argonne National Laboratory on November 5, 2003, a NEES Data/MetaData Task Group (DMD-TG) was formed to actively define and coordinate the data/metadata development tasks. A complete list of contributing members involved in this development is given in the Acknowledgements section of this report. The high level objective of this effort is to develop data models for earthquake engineering experiments and simulations. For this purpose sample data sets are being developed that demonstrate the features of the data models along with scenarios for the use of the data and models. Specifications for the tools necessary to support entering, importing, storing, searching, and extracting data from the repository are being proposed and developed.

1.2 Definition

One major task of the data/metadata effort is to develop a reference data model for supporting the major activities involved in earthquake engineering simulations. There are many existing data modeling techniques and tools that are available to help design and structure the data. A brief review of these relevant techniques and approaches has been reported earlier [23]. A data model is in essence a representation of the data and their interrelationship and provides a conceptual or implementation view of the data. A **data model** can be viewed as the “grammar”, “vocabulary” and “content” that represent the types of “information” stored in a “system”. The **grammar** defines the relationships among the data **elements** in the system; the **vocabulary** defines the terminology used to describe these elements; the **content** defines what is to be included in the system. The data model should be independent of hardware/software platforms so that its implementation can be universal.

Within the scope of the NEESgrid data/metadata effort, **data** is defined as *all* of the project related information and encompasses **observational** (or *acquired*) data recorded prior to the experiments and during the experiments by means of sensors, cameras and the like; **computational** (or *generated*) data generated as a result of modeling, simulations, post-processing; and **literature** in the form of reports, journal papers, drawings, etc. Associated descriptive and related data, i.e. **metadata** (or data about the data), are defined and published in a prescribed (by NEESgrid) format and language (as of this writing, the metadata is represented in OWL (Web Ontology Language) [20] format). It is expected that a set of functional system-wide services for storage, retrieval, and management of data and metadata associated with a project will be available as part of the NEESgrid infrastructure. These services will be based on specialized *data models* with only limited *content* populated by *elements* that are most critical to (1) the execution of a project, i.e. conduct and control of experiments and simulations; (2) the equipment, collection of sensor and video/image data, visualization; and (3) the storage, retrieval and management functions. However, it should be noted that the so-called limitations on the content and elements will not prevent future extensions of the data models and the integration of new project related elements in the NEESgrid infrastructure.

1.3 Scope

This document starts with a description of the NEESgrid data modeling approach and a brief review of several related data models. The overview and details of the developed reference NEESgrid data model are then presented. A brief summary is also provided to discuss the approach and findings in developing the reference data model for NEES experiments. This report serves to outline current tasks and approaches to define data models for supporting the activities involved in earthquake engineering experiments. The data model for supporting computer simulations has been described elsewhere in Ref [8]. Although the NEESgrid reference data model is intended to focus on the data requirements for shake table and centrifuge experiments, a large portion of the model should be of sufficient generality to be used for other types of experiments, such as pseudo-dynamic, Tsunami, or field tests.

2 Data Modeling Tool and Approach

There are many existing data modeling techniques and tools that are available to help design and structure the data. A review of data modeling formats, approaches, tools, and a few existing data models has been presented in Ref [23]. This section briefly discusses the tool selected for developing the NEESgrid data model and the basic concepts of object-oriented data modeling.

2.1 Protégé – Data Modeling Tool

There are many data modeling or software design tools that can be used to facilitate the design of a data model for a specific application. To facilitate the development of the NEESgrid data model, we selected

Protégé (<http://protege.stanford.edu>), which is an open-source software package designed to help develop ontology for knowledge-based systems [10]. Ontology represents explicit formal specifications of the terms in the domain and the relations among them [22]. As open source software, Protégé has attracted a wide variety of plug-ins from around the world to enhance its capabilities. Some of these software plug-ins allow a model developed in Protégé to be exported in many standard formats, including UML (Unified Modeling Language [3]), XML Schema (<http://www.w3.org/XML/Schema>), RDF (Resource Description Framework, <http://www.w3.org/RDF/>), OWL (Web Ontology Language, <http://www.w3.org/2001/sw/WebOnt/>), and others.

In Protégé, a graphical user interface (GUI) is provided to facilitate ontology development. The interface enables the modeling of an ontology of classes to describe a particular subject with a set of concepts and their relationships. The interface also allows direct entering of specific instances of data and the creation of a knowledge base. Figure 1 shows an example of the GUI, with the view of classes shown in the left window, and the view of detailed attributes of a class (e.g. Project class) shown in the lower right window.

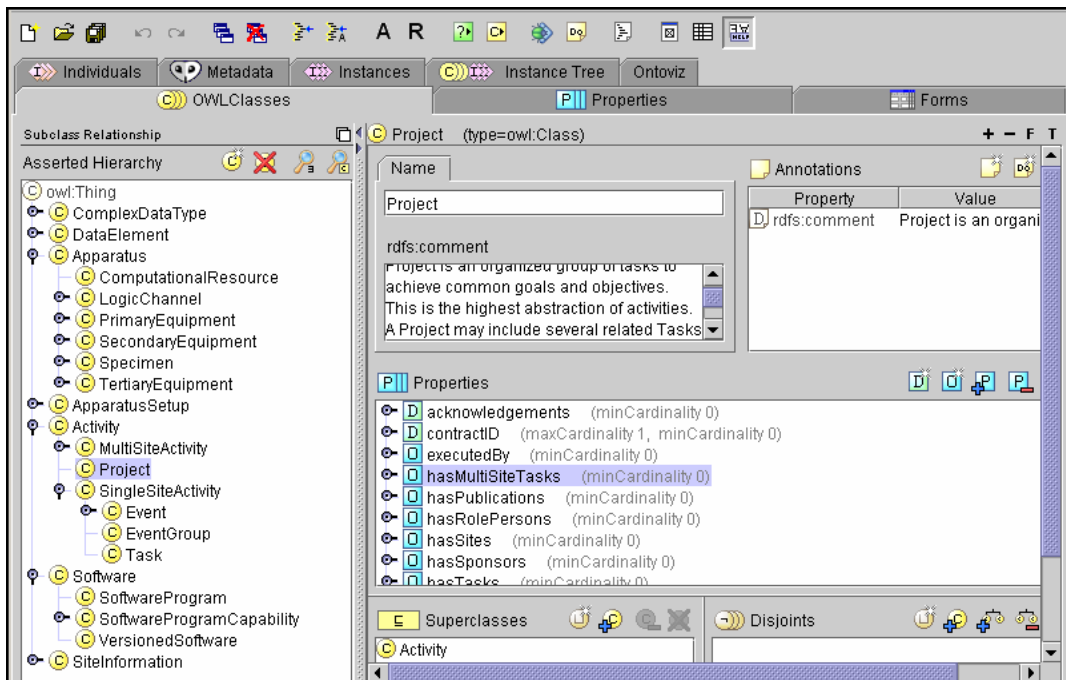


Figure 1 – Protégé Interface with OWL Pulgin

2.2 Object-Oriented Data Modeling

Object-oriented data modeling approach is employed in the development of NEESgrid data model. In an object-oriented data model, information is modeled as **objects**, which can be any sort of entities [27]. The general representation of a certain type of objects is called a **class**, which represents an explicit description of concepts in a domain. The creation of an object of a certain class is called **instantiation**. The relationship between an object and a class can be viewed analogically in a procedural language in that a variable is a particular instance of a pre-defined type such as an integer. For example, a *Project* is modeled as a class, and a MOST experiment [1] is an object instance of the *Project* class.

An object encapsulates certain related data as **slots**, which are also called **attributes** or **properties**. Slots can have different **facets** describing the value type, allowed values, the number of values

(**cardinality**), and other features of the values the slot can take. A value-type facet describes what types of values can fill in the slot. Some common value types are string, number, Boolean, enumerator, and object instance. Allowed objects and the variety of values (e.g. minimum number, maximum number) of a slot are often referred to as the **range** of a slot. Slot cardinality defines how many values a slot can have, such as single (at most one value) or multiple (more than one value). For example, “minute” is a slot of class *Time*, with data type defined as integer, cardinality one and range from 0 to 59.

An object connects with other related objects via some relationships. The relationship types commonly used include *classification*, *association*, *aggregation* and *generalization* [6, 28, 29]. These relationships types may in turn impose certain “object-oriented” features and integrity constraints to help maintain consistency and correctness of the data in the model. One important feature of object-oriented modeling is the concept of class hierarchies, with slots of a **superclass** being inherited by its **subclasses**. This **inheritance** feature allows us to define the common slots used by several classes at the highest possible level in the hierarchy, which avoids the duplication of slots at the lower levels. A class can have subclasses that represent concepts that are more specific than the superclass. For example, we can divide the class *Activity* into *Project*, *SingleSiteActivity*, and *MultiSiteActivity*. The class *SingleSiteActivity* in turn can be divided into *Task*, *EventGroup*, and *Event*. The common slots for all these *Activity* classes are name, description, start Time and end Time.

In object-oriented data models, a class can be **abstract** or **concrete** [22]. A concrete (or physical) class can have direct instances, as in the case that a MOST experiment is an instance of the class *Project*. On the other hand, abstract class cannot have any direct instances. For example, the *Activity* class is defined as the general abstraction of action or process, and thus a direct instance cannot be created.

3 Relevant Data Models

There have been several relevant developments that can potentially benefit the development of the data model for NEES experiments. The following discussion focuses on reviewing some of these models and their applicability for the NEESgrid data/metadata efforts.

3.1 Oregon State Model

Oregon State University and the Northwest Alliance for Computational Science and Engineering (NACSE) have developed a data model for describing laboratory tsunami experiments (<http://nees.orst.edu/IT/data.model/>) [2]. The model contains relationships among projects, experiments, researchers, equipment, experimental results, etc. The model is designed for storing the experimental data in a relational database. Figure 2 shows a high-level E-R diagram for the data model [2]. The diagram shows that a project may have multiple experiments, an experiment may have multiple configurations, a configuration may have multiple trials, etc. The model consists of a relatively small number of entities to make the structure of the model simple and to keep the number of tables manageable. A flexible scheme is used to assign attributes to entities. For instance, the Equipment table may include entries for any number of pieces of physical equipment used in an experiment – from strain gauges to wave basins – rather than requiring the assignment of each sensor or gauge to a particular slot. Another feature of the model is its extendibility; for instance, the model has the ability to incorporate new types of measurement instruments. Presently, the model is designed primarily for a tsunami wave basin experiment. The current model lacks details on some common elements, such as time, location, unit, and ground motion. However, the Oregon State model does provide many insights that are valuable for the development of other NEESgrid data models.

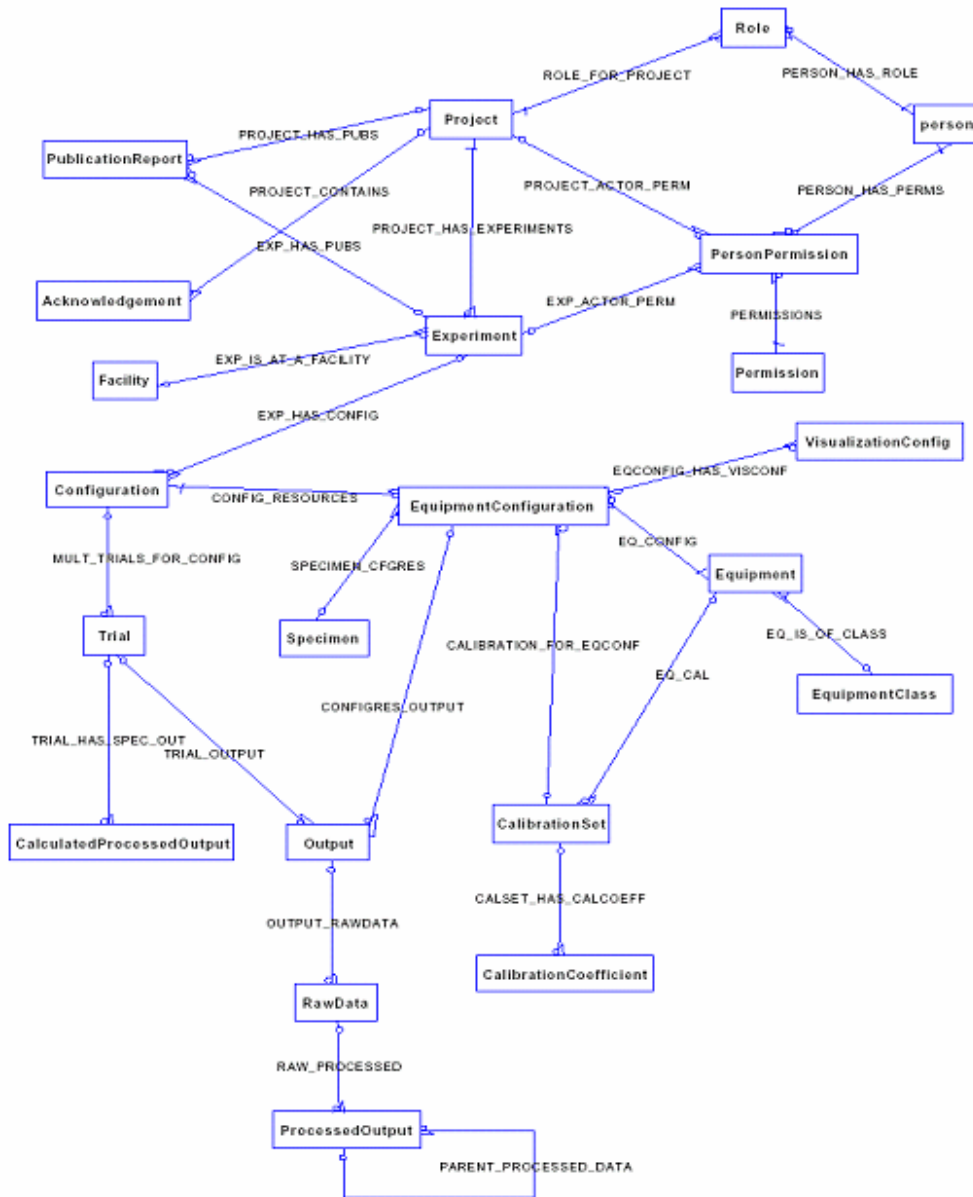


Figure 2 – E-R Diagram of the Oregon State Model [2]

3.2 Ontology of Science

There have been similar efforts in the Science community to develop standard models for science projects. An Ontology of Science, which is modified from the KA² ontology developed earlier by Knowledge Annotation Initiative of the Knowledge Acquisition Community [4], is available at http://protege.stanford.edu/ontologies/ontologyOfScience/ontology_of_science.htm.

The Ontology of Science is designed for modeling scientific events and educational events, such as a scientific conference, a research project, or a software development project. The ontology has a high-level project entity. As shown in Figure 3, the project has relationship with other entities, including people, organization, product and events. The project also has a start date and an end date, which can be used to calculate the duration of the project. Besides the high-level modeling, the Ontology of Science

also provides detailed modeling of several common elements, such as people, scientific documents, location, and time. The ontology provides a good organizational view that could be used to organize a collection of experimental projects.

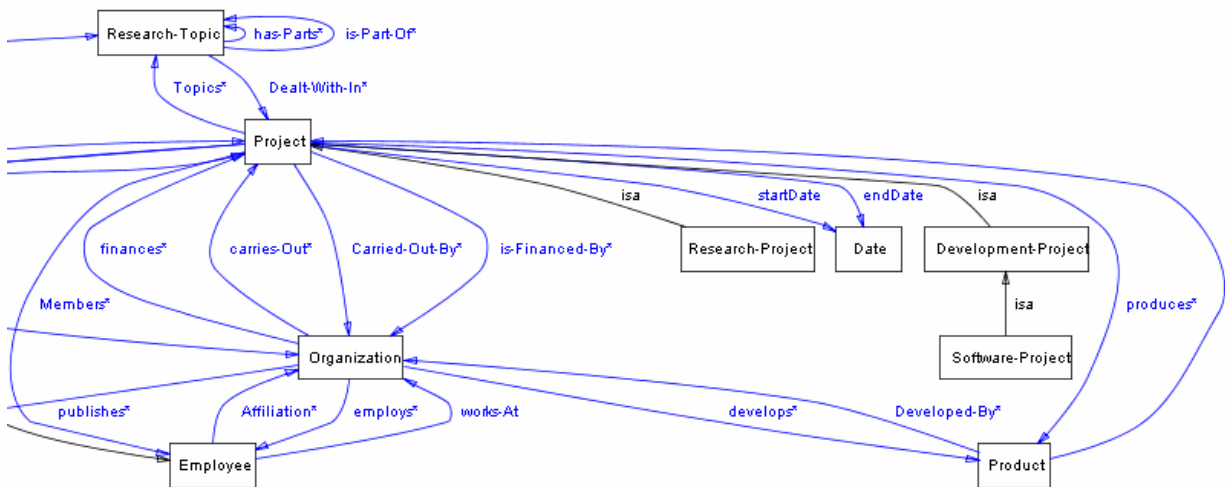


Figure 3 – Project Model of the Science Ontology

3.3 Berkeley CUREE/ Kajima

In the structural engineering domain, Prof. Bozidar Stojadinovic of University of California, Berkeley and his collaborators have created a framework for the integration and visualization of structural state data (see <http://www.ce.berkeley.edu/~boza/research/Curee-Kajima/>). The goal of this project is to conceptualize, develop and implement a framework for gathering, classifying and integrating structural data collected in and around a structure, and to enable effective visualization and fusion of such data to define the state of a structure.

Presently, the model focuses on the collection of observed data and comprehensive modeling of collector and data type. An abstract class, Collector, is a general object that collects or generates data about a structure. A Collector can be any type of sensor, a camera, numerical analysis, or even a person. Each type of collector is capable of collecting different types of data and handling different types of input and output data. The observed or generated data may include numerical data, descriptive textual data, or visual and graphical data (such as visual imagery from a camera, hand sketch, drawings, etc.).

The sensor model of Berkeley CUREE/Kajima provides a good representative class of different types of sensors commonly used for structural monitoring. Individual models of sensors are defined as separate objects to allow special calibration information and permits creation of separate instances of each sensor with individual serial numbers or characteristics. As shown in Figure 4, different types of sensors, such as accelerometers, strain gages, and thermocouples, are modeled as subclasses of Sensor. The model has been developed using Protégé and can produce different representation formats.

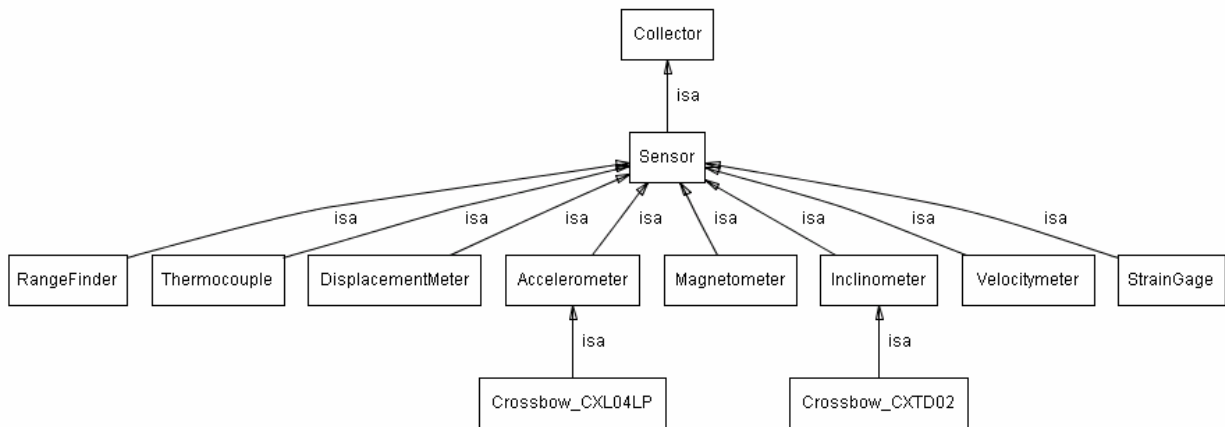


Figure 4 – Sensor Model of the Berkeley CUREE/Kajima

3.4 SensorML

Sponsored by the OpenGIS Consortium, SensorML (<http://vast.uah.edu/SensorML/>) is predisposed toward preserving part of the vital sensor data required for both real-time and archival observations [5]. The purpose of SensorML is to provide general sensor information, to support the processing and analysis of sensor measurements, to describe performance characteristics, and to archive properties and assumptions regarding the sensors.

SensorML provides an XML schema for defining the geometric, dynamic, and observational characteristics of a sensor. The root for all SensorML documents is Sensor, which represents a device for the measurement of physical quantities. Key components of a sensor model includes sensor identification, sensor location, constraints, platform attached by the sensor, coordinate reference system, sensor description, and measurement characteristics.

An important concept of SensorML is SensorGroup. A SensorGroup can be of two types, sensor package and sensor array. A sensor package is composed of multiple sensors that operate together to provide a collective observation or related group of observations. For example, a collection of sensors can be used in a combined fashion to create a sensor that measures, for example, temperature, strain, wave velocity and direction. A sensor array is a set of sensors of the same type at different locations. These locations may be within a single sensor frame, a different location on a single mount, or on different platforms. A sensor array produces observations that are used to build a spatial coverage.

SensorML provides a good prototype and a rich model for describing sensor information. It is particularly useful for in-situ experimental applications involving spatially distributed information.

3.5 Specimen Models

There have been many attempts to construct universal data (product) models to capture detailed descriptive information about building structures. The purpose of most of the existing models is to develop data exchange and sharing standards for CAD systems.

The CIMSteel Integration Standard (CIS/2) (<http://cic.nist.gov/vrml/cis2.html>) is the logical product model and electronic data exchange format for structural steel project information [7]. The CIS/2 standard is based on STEP (STandard for the Exchange of Product Data [12]) and is proposed to describe structures and engineering information, testing procedures and industry specific information. CIS/2 has been implemented for describing steel framed structures, from nuts and bolts to materials,

loads to frames and assemblies. Adopted by the American Institute for Steel Construction, many steel structure software packages now provide CIS/2 import and/or export capabilities.

Another industry standard related to building structure is the Industry Foundation Classes (IFC) (<http://www.eccnet.com/step/>) developed by a consortium, the International Alliance for Interoperability (IAI), to provide data exchange and sharing capabilities for the building and construction industry [11]. The IFC is a data representation standard and file format for defining architectural and constructional CAD graphic data. The IFC uses text-based structures for storing the definitions of objects encountered in the building industry.

While various domain models have been developed for the building industry, they are either too specific (CIS/2 for steel, for example) or too general (IFC for all phases of a building project) to be useful for describing a specimen or a test model typically used in earthquake engineering experiment and simulation. Although these models could share many insights to describe a structural component, they may not be useful for experimental test applications.

4 Overview of the Reference Data Model

As depicted in Figure 5, the NEESgrid data/metadata task group is working towards producing end-to-end solutions that integrate the site specifications database, the project level model, domain specific data models, and common elements. To capture all these data, the reference data model is designed to include six base classes, namely *SiteInformation*, *Activity*, *Apparatus*, *ApparatusSetup*, *DataElement*, and *ComplexDataType* [25]. The high-level class diagram of the reference data model is presented in Figure 6, which shows the association relationship among classes. (The *ComplexDataType* class, which is employed to support other base classes, is not shown in the figure.) The association relationship exists between classes when an object of one class *knows/contains* an object of another class. For example, a **Project** object knows about its **Tasks** objects, a **Project** also contains **Organizations**, **Sites**, and **RolePersons**. RolePerson is in turn defined as the combination of a Person and his/her role in a Project. The arrow in Figure 6 denotes the direction of the relationship *contains*; i.e., $A \rightarrow B$ indicates that class A *knows/contains* class B.

This section presents details of the six defined groups of base classes and some of their subclasses. In the discussions below, each class is described with a table that has the following four columns:

- **Slot Name:** which is the name associated with a slot of a class. Slot of a class is defined as the property describing a feature and/or an attribute.
- **Type:** which defines what type of values can fill in the slot. Slot type can be primitive (such as String, Integer, Float, and Boolean) or instance-type. The instance-type slots allow definition of relationships between individuals. Slots of instance-type must also define a list of allowed classes (started with a colon ‘:’ sign in the tables) from which the instances can come. For example, a slot `hasTasks` for the class `Project` may have instances of the class ‘:Task’ as its values.
- **Cardinality,** which defines how many values a slot can have. There are three types of cardinality used in the data model, including key value (must have one value, normally used to identify an object, denoted as ‘1’), single cardinality (allowing at most one value, denoted as ‘0:1’), and multiple cardinality (allowing any number of values, denoted as ‘0:*’).
- **Description:** which is documentation that describes the definition, meaning, and usage of a particular slot.

Due to the object-oriented feature of inheritance, all the slots of a superclass are inherited by its subclasses. For the sake of simplicity, the inherited slots are not listed for a subclass, but rather only the slots specific to the subclasses are presented. In the following, we will use the convention that class names are started with capital letters and the slots names are started with lower-case letters.

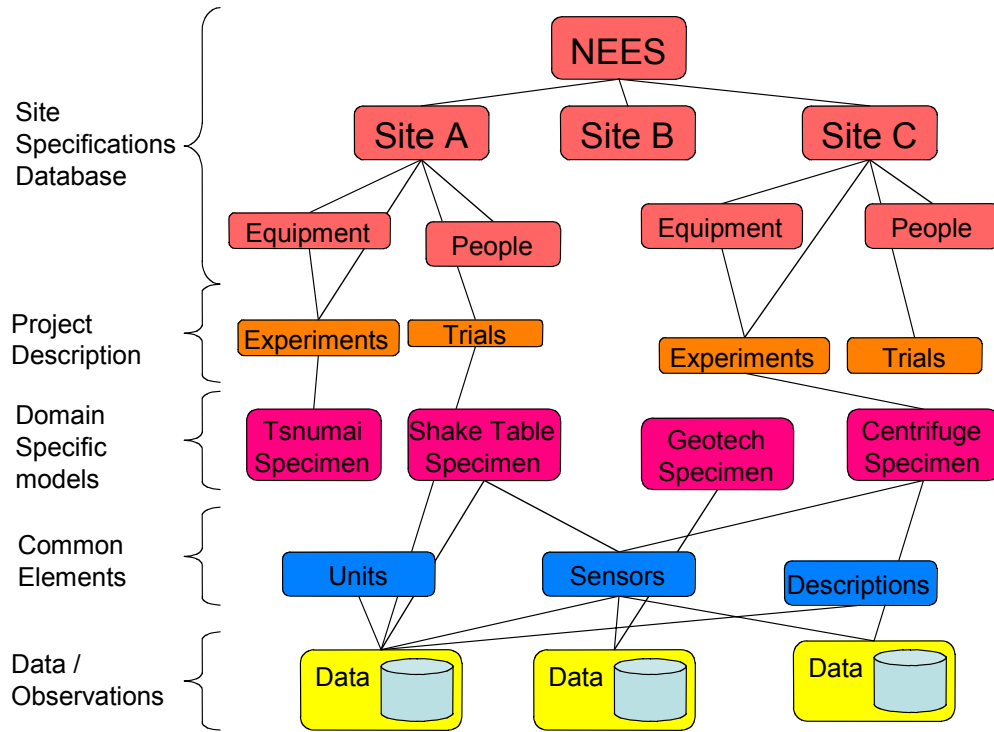


Figure 5 – Overall Data Model for NEESgrid (Courtesy of Chuck Severance)

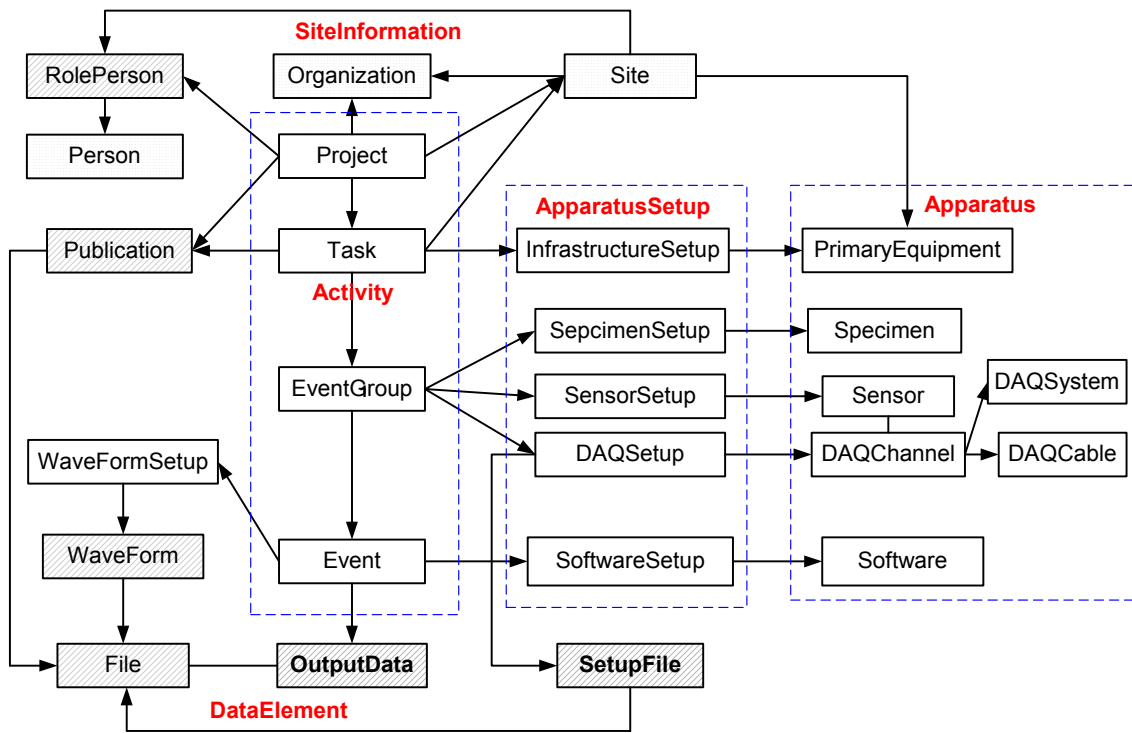


Figure 6 – High-level Class Diagram of the Reference Data Model

4.1 SiteInformation

A typical experiment site is hosted by certain Organization, and the site has personnel playing different roles, facilities, equipments and other information. The class hierarchy of SiteInformation is shown in Figure 7. This group of classes is intended to be associated with the site specifications database [13] that is currently under development by the NEES community. There are no properties (slots) defined for the base SiteInformation class.

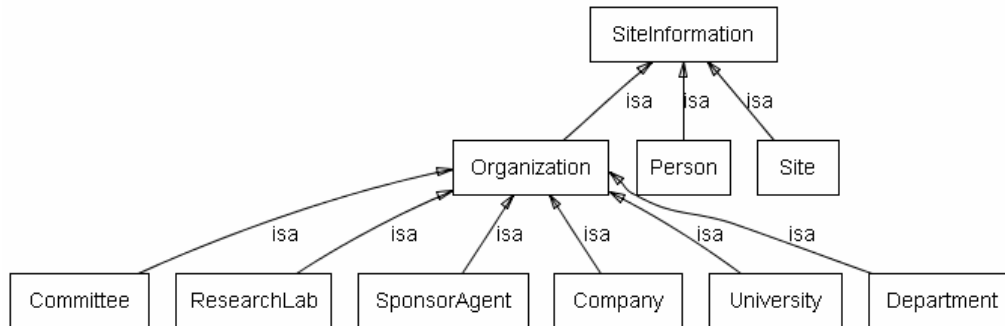


Figure 7 – Class Hierarchy of SiteInformation (generated by Protégé)

4.1.1 Organization

The Organization is defined as an administrative and functional unit, including the personnel of such a unit. Detailed description of the Organization class is shown in the following table.

Slot Name	Type	Cardinality	Description
name	String	1	The name of this particular Organization.
NEESCode	String	1	A unique reference code assigned by the NEES Consortium.
address	String	0:1	A place where an organization resides.
description	String	0:1	A simple description of the object.
homePage	String	0:1	Official web site of the Organization.

The subclasses of the Organization, namely **Committee**, **ResearchLab**, **SponsorAgent**, **Company**, **University**, and **Department**, share the same slots and properties as the Organization class. These subclasses are introduced in the reference data model to differentiate different types of Organizations.

4.1.2 Person

The Person class represents all of the personnel who participate in projects and experiments. Detailed description of the Person class is shown in the following table.

Slot Name	Type	Cardinality	Description
firstName	String	1	A given name or the name that occurs first in a given name.
lastName	String	1	Surname; a name shared in common to identify the members of a family.
title	String	0:1	A description or role of a person.
personID	String	1	Unique identification number assigned for a person.

phoneType	String	0:*	The type of phone or fax number of the Person.
phoneNumber	String	0:*	Digital telephone number or fax number.
address	String	0:*	Places where a Person works at or lives in.
email	String	0:*	Email addresses such as "name@location.org", used within a system for sending and receiving messages electronically over a computer network.
homepage	String	0:*	The URL of the homepage of the Person.
affiliation	:Organization	0:*	The Organization with which the Person is associated. If the Person is an employee, this would be the company that employs the Person.
otherInfo	String	0:*	Other information about the Person, such as his/her responsibilities to a Project.

4.1.3 Site

The Site class is modeled to represent large-scale facilities, such as the site that hosts shake table or centrifuge equipment. A Site is defined as having an inventory of devices, equipments, and personnel. The relationships of the Site class with other classes are shown in Figure 8. Detailed description of the Site class is shown in the following table.

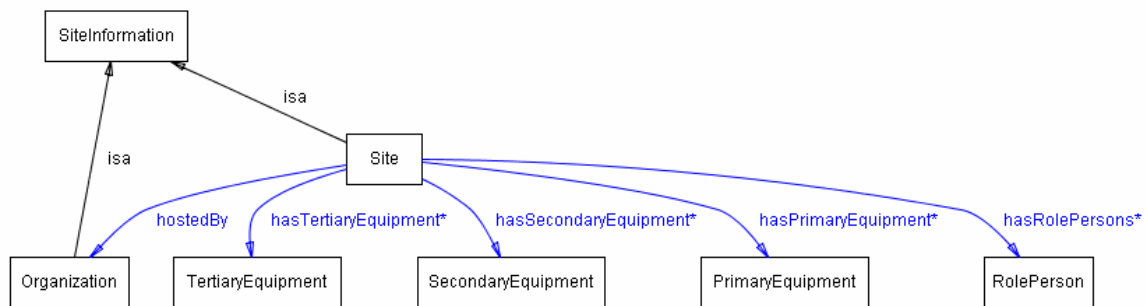


Figure 8 – Relationship of Site with Other Classes (generated by Protégé)

Slot Name	Type	Cardinality	Description
name	String	1	The name of a unique Site.
shortDescription	String	0:1	A short description of a Site.
longDescription	String	0:1	More detailed description of a Site.
hasPrimary Equipments	:Primary Equipment	0:*	A list of PrimaryEquipment belonged to the Site.
hasSecondary Equipments	:Secondary Equipment	0:*	A list of SecondaryEquipment belonged to the Site.
hasTertiary Equipments	:Tertiary Equipment	0:*	A list of TertiaryEquipment belonged to the Site.
hasRole Persons	:Role Person	0:*	A Site has a list of RolePersons, which represent Persons who play different roles for the Site.
hostedBy	:Organization	0:1	The organization where a Site is hosted by.

4.2 Activity

The Activity class is designed to support project level modeling. Common properties of Activity classes include objective, description, startDatetime, endDatetime, and others. The duration of an Activity can be calculated by using the startDateTime together with the endDateTime of an Activity. The startDateTime of an Activity can also be used to identify which Activity happens first, i.e. to sort out the sequence of several Activities. There are four basic types of Activities, including Project, Task, EventGroup, and Event. As shown in the class hierarchy of the Activity class (Figure 9), direct subclasses of the Activity class include Project, SingleSiteActivity, and MultiSiteActivity. Detailed description of the Activity class is shown in the following table.

Slot Name	Type	Cardinality	Description
name	String	1	The name of an Activity.
shortDescription	String	0:1	A short description of an Activity.
longDescription	String	0:1	More detailed description of an Activity.
local TimeZone	String	0:1	Local time zone in effect at the time the activity started.
startDateTime	DateTime	0:1	The beginning time of an Activity.
endDateTime	DateTime	0:1	The ending time of an Activity.

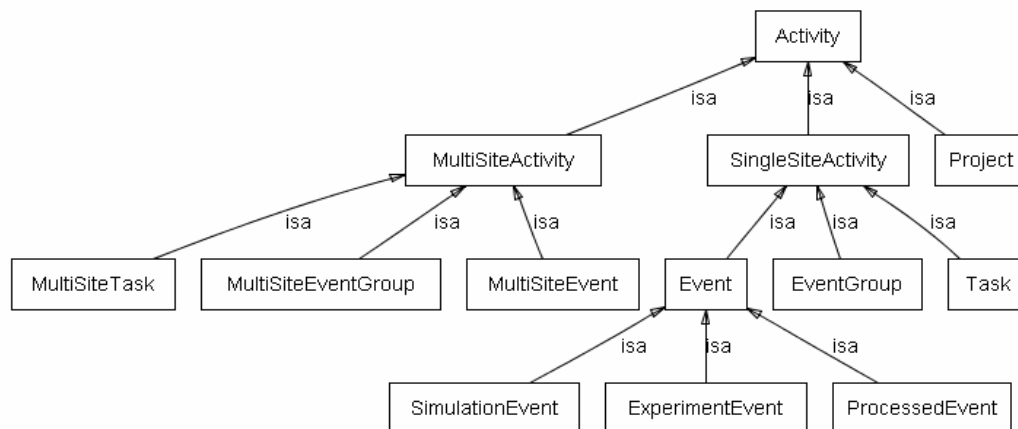


Figure 9 – Class Hierarchy of Activity (generated by Protégé)

4.2.1 Project

A Project is a collection (organized group) of related tasks carried out by certain organizations and designed to achieve specific goals. A Project can be sponsored by one or more funding sources. A Project includes one or more related Tasks. For example, the CUREE-Caltech Woodframe project was sponsored by CUREE (Consortium of Universities for Research in Earthquake Engineering), had many tasks/activities to study the performance of woodframe structures, and with the objective to reduce earthquake losses to woodframe construction (<http://www.curee.org/projects/woodframe/index.html>). Detailed description of the Project class is shown in the following table.

Slot Name	Type	Cardinality	Description
objectives	String	0:*	The objectives of a project.
contractID	String	0:1	NSF sponsorship number or other identification number for a NEES project contract.
NEESCode	String	1	A unique reference code assigned by the NEES Consortium for the Project.
keyWords	String	0:*	Key words and references for finding information.
Acknowledgements	String	0:*	Acknowledgements represent financial support, contributions of special instrumentations, or other types of sponsorship associated with the Project.
executedBy	:Organization	0:*	A list of organization(s) that execute the Project.
hasTasks	:Task	0:*	A collection of Tasks that associated with the Project.
has MultiSiteTasks	:MultiSiteTask	0:*	A list of MultiSiteTasks that associated with the Project.
hasSites	:Site	0:*	A list of Sites at which the project executed.
has Sponsors	:Organization	0:*	A Project has sponsors identified as Organizations.
has RolePersons	:RolePerson	0:*	A Project has identified RolePerson, which represents a Person plays a particular role in an Activity
has Publications	:Publication	0:*	A Project has Publications identified with them, and/or produced through the Project.

4.2.2 Task

A Task belongs to a particular Project and contains one or more EventGroups. Each Task typically serves a specific role in a Project. In case of an experiment, each Task has a distinct InfrastructureSetup. In other words, any changes to the InfrastructureSetup would initiate a new Task. For example, Task 1.1.1 of the CUREE-Caltech Woodframe project refers to the shake table test of a simplified two-story single-family house [9]. Detailed description of the Task class is shown in the following table.

Slot Name	Type	Cardinality	Description
keyWords	String	0:*	Key words and references for finding information.
has RolePersons	:RolePerson	0:*	A Task has several identified RolePersons.
has Publications	:Publication	0:*	A Task has Publications identified with them, and/or produced through the Task.
has EventGroups	:EventGroup	0:*	A Task has a list of EventGroups, which is defined as a collection of Events.
hasInfrastructure Setups	:Infrastructure Setup	0:*	A Task has a list of InfrastructureSetups. Any major changes to the InfrastructureSetup would initiate a new Task.

4.2.3 EventGroup

An EventGroup is defined as a collection of Events. Any change to the data acquisition setup, sensor setup or the specimen setup would initiate a new EventGroup. The sequence of EventGroups in a Task is determined by their startDateTime. For example, Test Phase 6 of Task 1.1.1 of the CUREE-Caltech Woodframe project is identified as an EventGroup because the test structure (specimen) has been modified after Test Phase 5 [9]. Detailed description of the EventGroup class is shown in the following table.

Slot Name	Type	Cardinality	Description
hasEvents	:Event	0:*	An EventGroup has a list of associated Events.
hasDAQSetups	:DAQSetup	0:*	An EventGroup has a list of DAQSetups, which represent the setup of electronic devices whose primary purpose is to acquire data.
hasSensor Setups	:Sensor Setup	0:*	An EventGroup has a list of SensorSetups, which define the setup of Sensors, and the setup of Sensors with respect to Specimen.
hasSpecimen Setups	:Specimen Setup	0:*	An EventGroup has a list of SpecimenSetups, which define the setup of Specimen with respect to PrimaryEquipment.

4.2.4 Event

An Event, which is the atomic level of Activity, refers to each single run of an experiment or a simulation. Events within an EventGroup may have different input motions, loading protocols, etc. For each Event, certain outputs, such as sensor readings or simulation results, are generated and recorded. The sequence of Events in an EventGroup is determined by their startDateTime. Three types of Event are defined in the model, namely ExperimentEvent, ProcessedEvent and SimulationEvent. An example ExperimentEvent is a particular test within Test Phase 6 of Task 1.1.1 of the CUREE-Caltech Woodframe Project [9]. Detailed description of the Event class is shown in the following table.

Slot Name	Type	Cardinality	Description
testType	String	0:1	Types of tests such as shake table, centrifuge, tsunami, reaction wall, various field tests, and etc.
hasWave FormSetups	:WaveForm Setup	0:*	An Event has a list of WaveFormSetups, which are essentially the input.
hasOutput Data	:Folder	0:*	An Event has output data, which includes raw or processed data. The output can be organized in Folders.

4.2.5 MultiSiteActivity

MultiSiteActivity is a collection of Tasks, EventGroups and Events that may be carried out at more than one Site. MultiSiteActivity is used to establish the link among related Activities, which may be carried out at different Site but serve to achieve a common goal. MultiSiteActivity class has several subclasses, including MultiSiteTask, MultiSiteEventGroup, and MultiSiteEvent. The details of MultiSiteActivity class and its subclasses are shown in the following table.

Slot Name	Type	Cardinality	Description
MultiSiteActivity			
hasSite	:Site	1:*	A list of Sites where all the MultiSiteActivities are carried out.
MultiSiteTask is defined as a collection of Tasks, which are related and may be carried out at different Sites. MultiSiteTask contains one or more MultiSiteEventGroups.			
hasTasks	:Task	0:*	A MultiSiteTask has a list of Tasks associated with it.
hasMultiSiteEventGroups	:MultiSiteEventGroup	0:*	A MultiSiteTask has a list of MultiSiteEventGroups.
MultiSiteEventGroup is defined as a collection of EventGroups which are related and may be carried out at different Sites. MultiSiteEventGroup contains one or more MultiSiteEvents.			
hasEventGroups	:EventGroup	0:*	A MultiSiteEventGroup has a list of EventGroups. EventGroup is defined as a collection of Events.
hasMultiSiteEvents	:MultiSiteEvent	0:*	A MultiSiteEventGroup has a list of MultiSiteEvents.
MultiSiteEvent is defined as a collection of Events which are related and may be carried out at different Sites. The Event can be an experiment or a simulation.			
hasEvents	:Event	0:*	A MultiSiteEvent has a list of Events. Event is defined as one run of an experiment or a simulation.

4.2.6 Layout of Activity Classes

The reference data model explicitly models certain Activities that are carried out at multiple Sites. Figure 10 shows an example project that has a single site Task (e.g. Task1) and a MultiSiteTask (e.g. M_Task1). The MultiSiteTask M_Task1 has Tasks that are undertaken at both Site1 and Site2. The MultiSiteEvent M_E1 has an Event E2 at Site1 and an Event E4 at Site2, and the MultiSiteEvent M_E2 has an Event E3 at Site1 and an Event E5 at Site2. As shown in Figure 10, although Project does not directly *contain* Task2 that takes place at Site2, Task2 can still be accessed from the Project since M_Task1 *contains* Task2. This design enables the support of the types of experiments (such as the MOST experiment [1]) that are carried out either simultaneously or independently at several Sites.

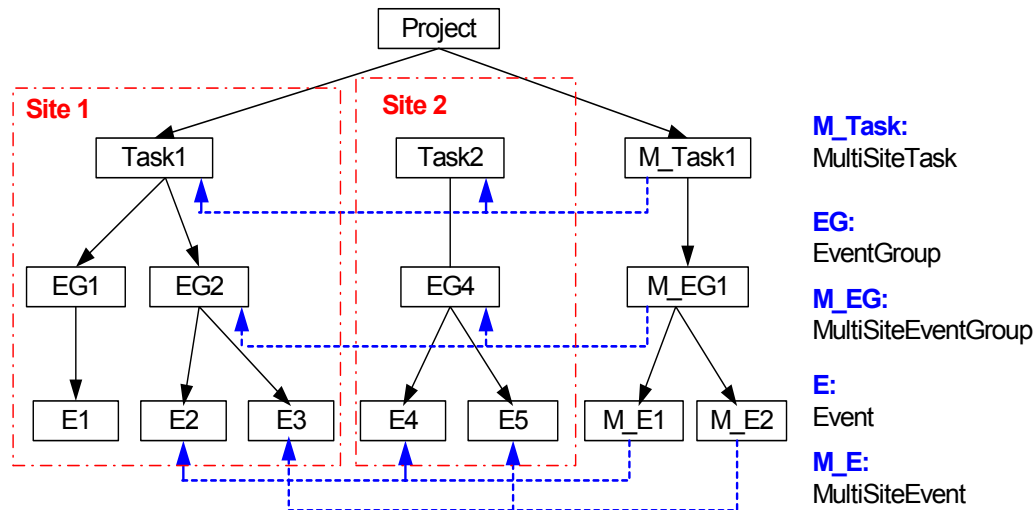


Figure 10 – Layout of an Example Project

4.3 Apparatus

Apparatus is defined as any equipment, specimen, or computational resource that may be used in an Activity. In the current version of the reference data model, the direct subclasses of Apparatus include Specimen, PrimaryEquipment, SecondaryEquipment and TertiaryEquipment. Explicit modeling of Specimen is not considered in the reference model [23]. Instead, only the most basic modeling is provided (as a collection of descriptive files, drawings, and/or photos). This design reflects current approach used to describe specimen in earthquake engineering experiments. However, the Specimen class can be extended to support other, more detailed, models. Detailed description of the Apparatus class is shown in the following table.

Slot Name	Type	Cardinality	Description
name	String	1	The name of an Apparatus.
shortDescription	String	0:1	A short description of an object.
longDescription	String	0:1	More detailed description of an Apparatus.
hasInfoFolder	:Folder	0:*	The Folder that contains all the related information (in the forms of files) of an Apparatus.

4.3.1 PrimaryEquipment

PrimaryEquipment is the major equipment that is used for the execution of an experiment with respect to a specific research area. As shown in Figure 11, direct subclasses of PrimaryEquipment are ShakeTableEquipment, CentrifugeEquipment, WaveBasinEquipment, FieldTestEquipment, and LargeScaleTestEquipment. Detailed description of individual PrimaryEquipment will be contained in the site specifications database [7]. Other types of PrimaryEquipment can be added to the data model as needed.

The current description of the PrimaryEquipment class is shown in the following table. These properties are shared by all its subclasses. More sophisticated modeling of different types of PrimaryEquipment can be added to the data model if necessary.

Slot Name	Type	Cardinality	Description
-----------	------	-------------	-------------

manufacturer	String	0:1	Entity that manufactured the apparatus.
operators	Person	0:*	Persons who operate the PrimaryEquipment
hasFigures	:VisualFile	0:*	A PrimaryEquipment has a list of Figures, such as Photos and Drawings.

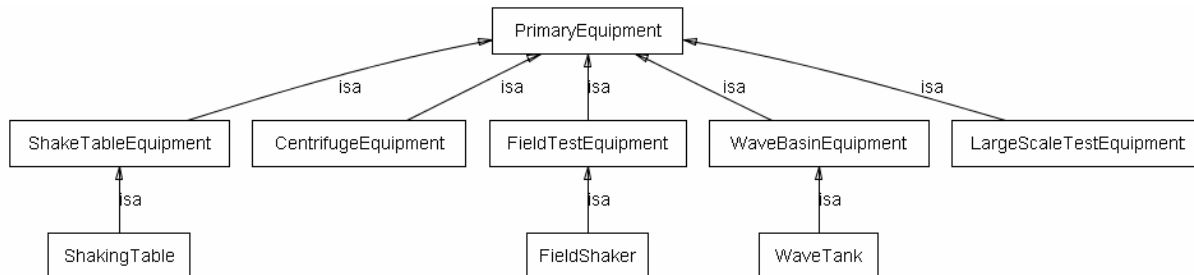


Figure 11– Class Hierarchy of PrimaryEquipment (generated by Protégé)

4.3.2 SecondaryEquipment

SecondaryEquipment may be a component of the PrimaryEquipment or may be a piece of equipment that facilitates the execution of an Event, data collection, and/or observation. Detailed description of the SecondaryEquipment class is shown in the following table.

Slot Name	Type	Cardinality	Description
owner	:Organization	0:1	The organization that the SecondaryEquipment belongs to.
manufacturer	String	1	The manufacturer of the apparatus.
serialNumber	String	1	Part (serial) number of the apparatus, usually given by the manufacturer.
manufacturer ModelNumber	String	1	Model number that uniquely identifies the manufacturer.
manufacturer PartNumber	String	1	Part number that identifies apparatus; this number is unique to the manufacturer
hasFigures	:VisualFile	0:*	A SecondaryEquipment has a list of Figures, such as Photos and Drawings.

The class hierarchy of SecondaryEquipment is shown in Figure 12. The direct subclasses of the SecondaryEquipment class include Sensor, ControlSystem, TelepresenceDevice, DAQDevice, HydraulicActuator, and ShakingSystem. Other types of SecondaryEquipment can be added to the data model as needed.

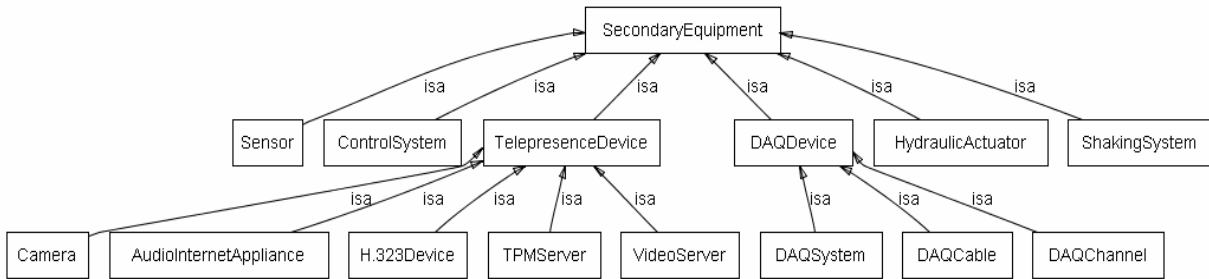


Figure 12 – Class Hierarchy of SecondaryEquipment (generated by Protégé)

One important type of SecondaryEquipment is the equipment and sensors used for data acquisition. Data schemas for describing sensors are available – one example is the SensorML [5] developed by OpenGIS Consortium. The reference data model includes a sensor model that is designed specifically to support earthquake engineering experiments. The data acquisition equipment is modeled as a collection of classes, including Sensor, DAQCable, DAQChannel, and DAQSystem. Figure 13 shows the relationships and the slots of these classes. Typically a data acquisition system involves at least three main components: (1) the sensors which respond to a physical stimulus and generate analog voltage signals; (2) a DAQchannel (a.k.a. signal conditioner as part of a DAQSystem) which receives the signal and uses predefined filter, gain, offset, excitation, sensitivity (calibration) information for Analog-to-Digital (A/D) and Engineering Unit (EU) conversions; and (3) a PC unit which uses some communications link (serial port, phone modem, radio modem, etc.) to retrieve the data. It is noted that A/D hardware can be either external to or part of the signal conditioner.

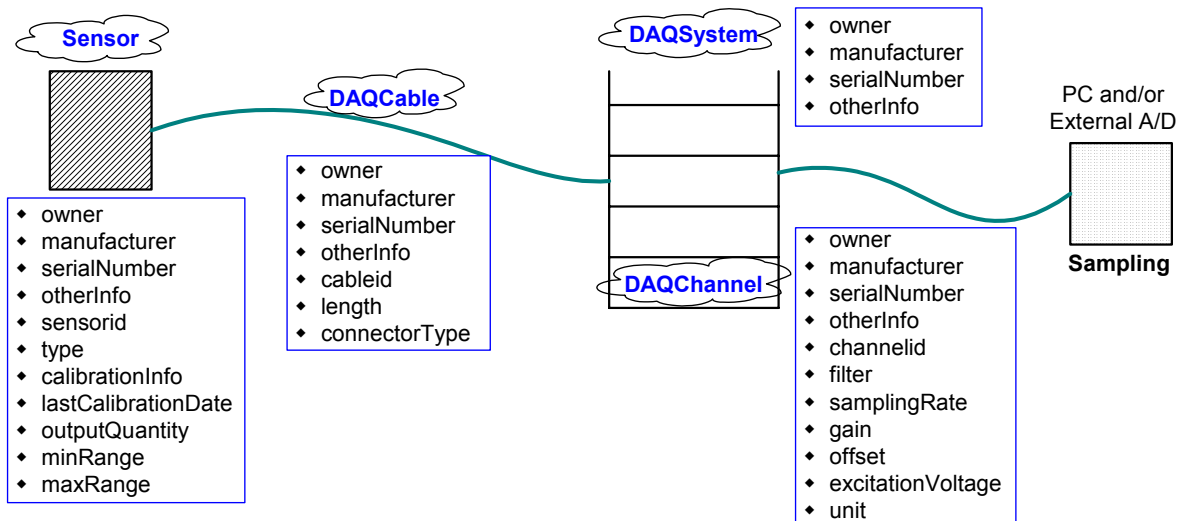


Figure 13 – Setup and Modeling of DAQ Devices

▪ **Sensor**

Sensor represents a device for the measurement of physical quantities. Key components of sensor modeling include sensor identification, sensor location, constraints, platform attached by the sensor, coordinate reference system, sensor description, and measurement characteristics. Detailed description of the Sensor class is shown in the following table.

Slot Name	Type	Cardinality	Description
sensorType	String	0:1	Information sensed by the Sensor, e.g. acceleration, pressure, displacement, strain, temperature, and etc.
minRange	:Float Measurement	0:1	The minimum measurable quantity.
maxRange	:Float Measurement	0:1	The maximum measurable quantity.
calibrationInfo	:File	0:1	A file that contains calibration information, expressed as formula, table, etc.
lastCalibration Date	DateTime	0:1	The date of last calibration.
outputQuantity	String	0:1	Quantity that sensor puts out in response to the input, it can be voltage, current, charge, or human read.
additional Specifications	String	0:*	Any additional information about the Sensor, e.g. dimension, weight, etc.

▪ DAQChannel

A DAQ (Data Acquisition System) channel is a digital computing means that accepts as its input a set of digital signals from which it generates as its output a second set of digital signals. DAQChannel collects signals from a connected Sensor, processes and transforms the signals, and transfers the signals to a DAQSystem for recording. Detailed description of the DAQChannel class is shown in the following table.

Slot Name	Type	Cardinality	Description
channel Location	:DAQSystem	0:1	The DAQ (Data Acquisition System) System that the DAQChannel connects to.
channelFilter	String	0:1	Filter type of the DAQChannel, such as Butterworth.
channelGain	Float	0:1	An increase in signal power, voltage, or current by an amplifier, expressed as the ratio of output to input. Also called amplification.
channelUnit	:Unit	0:1	A unit of measure of the DAQ (Data Acquisition) channel.
channel Offset	:Float Measurement	0:1	The offset value (measurement) of the DAQChannel.
excitation Voltage	:Float Measurement	0:1	The voltage measurement of the excitation current.
hasSensor	:Sensor	0:1	A Sensor that the DAQChannel is connected to.
hasDAQCable	:DAQCable	0:1	The cable that connects a Sensor to a DAQ (Data Acquisition System) Channel.
samplingRate	:Float Measurement	0:1	The frequency of sampling per unit time; the number of data points per unit time that a DAQ channel records data.

4.3.3 TertiaryEquipment

TertiaryEquipment is modeled as other general laboratory infrastructure details. The class is used for certain information requested by the NEES Site Specification Database. In the current version of the

data model, subclasses of TertiaryEquipment include ElectricalPower, HydraulicPower, and LAN. More detailed TertiaryEquipment modeling can be added to the data model if needed.

4.3.4 Specimen

Specimen is a primary component of the data model. A Specimen is one of many Apparatus used in conducting an Activity. One or more Specimens may be created for a Project. An Event or EventGroup is conducted utilizing the Specimens. Detailed description of the Specimen class is shown in the following table.

Slot Name	Type	Cardinality	Description
hasDescriptions	:Descriptive File	0:*	A Specimen has narrative descriptions and notes, which can be represented as DescriptiveFiles.
hasFigures	:Visual File	0:*	A Specimen has a list of Figures, such as Photos and Drawings.
hasSensorSetups	:SensorSetup	0:*	A specimen has a list of associated SensorSetups, which define the setup of Sensors with respect to Specimen.

4.4 ApparatusSetup

Universal modeling of the arrangement and setup of apparatus for all experiments is very difficult if not impossible. Not only are there different types of experiments (such as shake table, pseudo-dynamic tests, centrifuge, and tsunami) and different materials (such as concrete, steel, wood, etc.), but also the geometry of the specimen, the arrangement of sensors, and the configuration of PrimaryEquipment may be too complicated and cumbersome to model. For example, the “as-built” locations of sensors may be different from the prescribed “design” locations, and the precise physical locations (i.e., the coordinate x, y, z values) of sensors are often difficult to record. Therefore, it is recommended that the development of ApparatusSetup model be focused on tools and methodologies that can capture and organize CAD drawings, sketched drawings and notes, photos, narrative descriptions, electronic notes, etc.

The class hierarchy of ApparatusSetup in the current reference data model is shown in Figure 14. The InfrastructureSetup models the assembly and arrangement of the PrimaryEquipment used for a specific Task; any changes in InfrastructureSetup would trigger the launch of a new Task. The SpecimenSetup deals with the information on how the specimen is set up with respect to PrimaryEquipment. The SensorSetup includes the arrangement (location, orientation, etc.) of Sensors used in an experiment. The DAQSetup models the physical and electrical setup of one or more devices whose primary purpose is to acquire data. Any major change to SpecimenSetup, SensorSetup, or DAQSetup initiates a new EventGroup. The InputDataSetup deals with the choice and organization of input data to an Event. Any change to a new InputDataSetup indicates the beginning of a new Event.

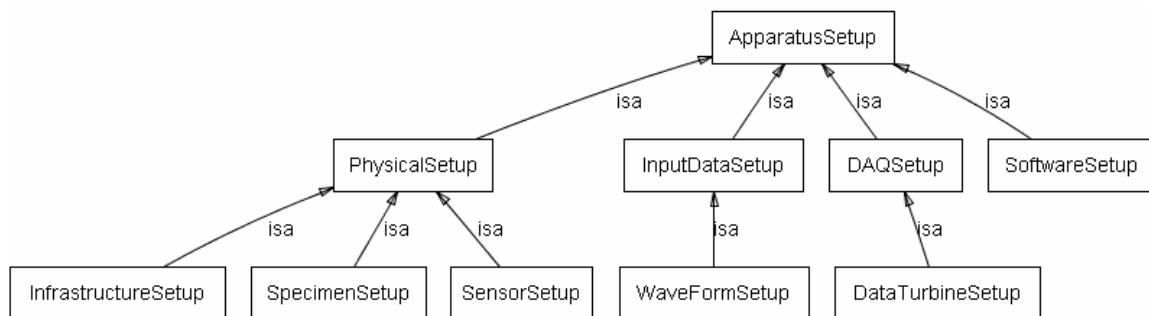


Figure 14 – Class Hierarchy of ApparatusSetup (generated by Protégé)

The ApparatusSetup class defines three slots, which are shown in the following table.

Slot Name	Type	Cardinality	Description
name	String	1	The name of the ApparatusSetup.
shortDescription	String	0:1	A short description of the ApparatusSetup.
longDescription	String	0:1	More detailed description of the ApparatusSetup.

4.4.1 PhysicalSetup

The PhysicalSetup models the physical setup of Apparatus, including related infrastructure, specimen(s), and sensor(s). The following table presents the detailed description of the PhysicalSetup class and its subclasses, including InfrastructureSetup, SensorSetup, and SpecimenSetup.

Slot Name	Type	Cardinality	Description
PhysicalSetup: The physical setup of the related Apparatus, including related infrastructure, specimen(s), and sensor(s).			
setup Descriptions	:Descriptive File	0:*	The text descriptions or notes that describe the setup.
setupFigures	:VisualFile	0:*	The figures and drawings that illustrate the setup. Locations (URI) to these image files need to be provided.
InfrastructureSetup: The assembly and arrangement of the PrimaryEquipment and certain SecondaryEquipment used for a specific Task. General laboratory infrastructure is described under TertiaryEquipment and will not be modeled in InfrastructureSetup.			
hasPrimary Equipments	:Primary Equipment	0:*	Defines the PrimaryEquipments that are associated with the InfrastructureSetup.
hasSecondary Equipments	:Secondary Equipment	0:*	Defines the SecondaryEquipments that are associated with the InfrastructureSetup.
SensorSetup: The arrangement (location, direction, etc.) of Sensors used in an EventGroup.			
hasSensors	:Sensor	0:*	A SensorSetup sets up multiple Sensors.
hasSensor Locations	:Apparatus Location	0:*	Indicates the sensor and its associated location. The location is represented in a coordinate system, which can be either global or local.
SpecimenSetup: The setup of specimen is modeled by this class. It is focused on how the specimen is set up with respect to PrimaryEquipment.			
hasPrimary Equipments	:Primary Equipment	0:*	A list of PrimaryEquipments that are used for the setup of Specimens.
hasSpecimens	:Specimen	0:*	A SpecimenSetup defines the setup of a list of Specimens.

4.4.2 DAQSetup

DAQSetup represents the setup of one or more electronic devices whose primary purpose is to acquire data. It can be simple or complex, depending upon the needs. Typically a data acquisition system involves at least three main components. First, sensors respond to a physical stimulus and transmit signals or change electrical property such as resistance. Second, a datalogger measures the electrical

signal, converts it to a number and stores either that value or some statistics on that value (average, maximum, minimum, standard deviation, etc.). Third, a PC uses some communications link (serial port, phone modem, radio modem, etc.) to retrieve the data from the datalogger. Detailed description of the DAQSetup class is shown in the following table.

Slot Name	Type	Cardinality	Description
hasDAQ Channels	:DAQ Channel	0:*	DAQSetup has a list of DAQChannels, which collect signals from a connected Sensor, process and transform the signals, and transfer the signals to a DAQSystem for recording.
hasFigures	:VisualFile	0:*	DAQSetup has a list of Figures, such as Photos and Drawings.
hasSetupFiles	:Numerical File	0:*	DAQSetup has a list of NumericalFiles that define the setup of DAQ Devices.

4.4.3 InputDataSetup

The InputDataSetup models the input signals to an ExperimentEvent. One type of input signal is waveform, which may be the record of a previous earthquake event or synthetic signals. A subclass of the InputDataSetup class is WaveFormSetup, which models the setup of a waveForm for specified Event(s). Detailed description of the WaveFormSetup class is shown in the following table.

Slot Name	Type	Cardinality	Description
hasDirection	:Location	1	WaveFormSetup has a defined direction from which the Wave was generated. The direction may be relative (to some part of the experiment apparatus) or measured using a coordinate system.
hasWaveForm	:WaveForm	1	A WaveFormSetup has a specified WaveForm.
scaleFactor	Float	1	A real number that is used to modify the original value of a quantity.

4.5 DataElement

DataElement represents all types of data that may serve as the input or be generated/processed during an Activity. The DataElement normally serves as Input/Output to an Activity. Types of DataElement include text document, publication, earthquake record, photo, CAD drawing, movie, etc. In the NEESgrid data/metadata effort, it is assumed that the data is saved in or translated into computer-readable format. Therefore, a DataElement object is represented in the format (such as a file) that can be saved in computer memory, on disks, or in some kind of data storage repository. Figure 15 presents part of the class hierarchy of the DataElement class. Several subclasses of the Publication class are not shown in Figure 15 for the purpose of keeping the figure readable. There are no properties (slots) defined for the DataElement class.

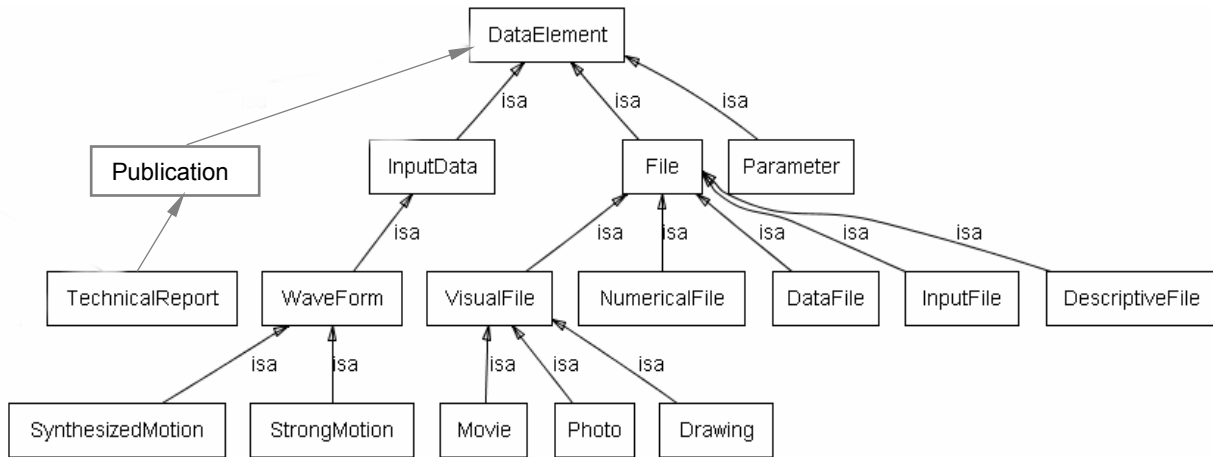


Figure 15 -- Class Hierarchy of DataElement (generated by Protégé)

4.5.1 Publication

Subclasses within this class relate to any scientific document published in a conference or by a research institution, such as Book, Journal, Proceedings, TechnicalReport, Thesis, and etc. More types of Publications can be added later if needed. Detailed description of the Publication class is shown in the following table.

Slot Name	Type	Cardinality	Description
title	String	1	A descriptive or general heading (as of a chapter in a book) of a Publication
authors	:Person	0:*	The Persons that originate, write, and/or create the Publication.
year	String	1	The year when the Publication was published.
keyWords	String	0:*	Words used as a reference point for finding other words or information.
URI	String	0:1	Indicates the location or identifier of a file object at which the Publication resides.

4.5.2 File

This class represents an object that can be saved in memory, on disks, or in the repository. A file can be saved in any computer-readable format. Subclasses of the File class include NumericalFile, VisualFile, InputFile, DataFile, and DescriptiveFile. Detailed description of the File class is shown in the following table.

Slot Name	Type	Cardinality	Description
name	String	1	The name of the File.
description	String	0:1	A simple description of the object.
URI	String	1	Indicates the location or identifier of a file object.

4.5.3 InputData

InputData is defined as the control signals to the Apparatus. A wavelet or waveform is built up of many sinusoidal waves of varying frequency and amplitude. The variation in frequency and amplitude of a particular wavelet can be shown as a frequency spectrum. By assuming a waveform in a model for the earth, an artificial seismic reflection record can be manufactured. A one-dimensional synthetic seismogram is formed by simply convolving a waveform with a reflection coefficient.

StrongMotion is a subclass of the InputData class. The StrongMotion class represents input strong ground motions. For future implementation, it can be linked to the existing strong ground motion databases. Detailed description of the StrongMotion class is shown in the following table.

Slot Name	Type	Cardinality	Description
name	String	1	The name of the InputData.
description	String	0:1	A simple description of the InputData.
motionRecord	:Numerical File	1	The strong motion record, usually organized as columns of real numbers.
recordUnit	:Unit	0:1	The Unit in which a time series record is presented.
samplingRate	:Float Measurement	0:1	The frequency of sampling per unit time; the number of data points per unit time that records StrongMotion data.
scaleFactor	Float	0:1	A real number that is used to modify the original value of a quantity.
peak Acceleration	:Float Measurement	0:1	Maximum absolute value of acceleration that is recorded or that can be achieved by an apparatus. The peak acceleration or rate of change of velocity with respect to time in a specified time series record.
peak Velocity	:Float Measurement	0:1	Maximum absolute value of velocity that is recorded or that can be achieved by an apparatus.
peak Displacement	:Float Measurement	0:1	Maximum absolute value of displacement that is recorded or that can be achieved by an apparatus.

4.6 ComplexDataType

ComplexDataType is defined in the reference data model to represent any data type that is not a simple data type such as integer, float, boolean, or character string. There are no properties (slots) defined for the ComplexDataType class. In the current version of the reference data model, the following ComplexDataType are provided.

4.6.1 Folder

This class represents a named or designated location where an object (DataElements, including Files, InputData, and Publications) can be saved in memory, on disks, or in the repository, in any computer-readable format. Detailed description of the Folder class is shown in the following table.

Slot Name	Type	Cardinality	Description
name	String	1	The name of the Folder.
description	String	0:1	A simple description of the Folder.
hasFiles	:DataElement	0:*	A Folder has a list of DataElements, which can be File, InutData, Parameter or Publication.

4.6.2 RolePerson

This ComplexDataType is defined to represent a particular role that a person plays in an Activity. Possible roles include principle investigator, co-investigator, research associate, postdoctoral assistant, graduate assistant, undergraduate student, technician, and etc. Detailed description of the RolePerson class is shown in the following table.

Slot Name	Type	Cardinality	Description
hasPerson	:Person	1	A RolePerson has a Person.
hasRole	String	1	RolePerson has a role within a project, such as principle investigator, co-investigator, research associate, postdoctoral assistant, graduate assistant, undergraduate student, technician, and etc.
isActive	Boolean	0:1	Indicator as to whether a Person is active or inactive, related to a specified project.

4.6.3 Unit

A precisely specified quantity in terms of which the magnitudes of other quantities of the same kind can be stated. (Note: A good way to model unit is yet to be determined.) A detailed description of the Unit class is shown in the following table.

Slot Name	Type	Cardinality	Description
name	String	1	The name of the Unit.
description	String	0:1	A simple description of the Unit.
conversion	Float	0:1	The conversion rate from one Unit to another. For instance, a foot equals to 12 inches.
relativeTo	:Unit	0:1	A way to describe a value, as relative to another particular Unit.

4.6.4 Measurement

The Measurement class represents the dimensions, capacity, or amount of something ascertained by measuring a measured quantity. A subclass of the Measurement class is FloatMeasurement, which is the Measurement described using real number(s), single-precision floating point. Float is a primitive data type (size/format: 32-bit IEEE 754). A detailed description of the FloatMeasurement class is shown in the following table.

Slot Name	Type	Cardinality	Description
value	Float	1	A numerical quantity that is assigned or is determined by calculation or measurement.
hasUnit	:Unit	1	Defines a specified Unit of measure.

4.6.5 DateTime

The time at which an event occurs, recorded as required. DateTime is externally represented as year, month, day, hour, minute, second, millisecond, etc., and is internally saved as a long integer. A detailed description of the DateTime class is shown in the following table.

Slot Name	Type	Cardinality	Description
year	Integer	1	A period of approximately the duration of a calendar year.
month	Integer	1	A measure of time corresponding nearly to the period of the moon's revolution and amounting to approximately 4 weeks or 30 days or 1/12 of a year.
day	Integer	1	The 24-hour period during which the earth completes one rotation on its axis.
hour	Integer	1	The time of day determined on a 24-hour basis.
minute	Integer	1	A unit of time equal to one sixtieth of an hour, or 60 seconds.
second	Integer	1	The 60th part of a minute of time.
millisecond	Integer	1	One thousandth of a second.

4.6.6 Angle

The Angle is the measurement of an angle or of the amount of turning necessary to bring one line or plane into coincidence with or parallel to another. A detailed description of the Angle class is shown in the following table.

Slot Name	Type	Cardinality	Description
degreeType	String	1	The hemisphere (N, E, S, W) is attached to the degrees as an attribute. Note: the values of '+' for N and E, and '-' for S and W are also permitted.
degree	Integer	1	A unit of latitude or longitude, equal to $\frac{1}{360}$ of a great circle.
minute	Integer	1	A unit of angular measurement equal to one sixtieth of a degree, or 60 seconds.
second	Integer	1	The 60th part of an angular measurement.

4.6.7 Location

The geometry/location is needed for finding sensor location, representing a specimen model, etc. The spatial location is currently modeled as the values in a coordinate system (i.e., x, y, z values). It should be noted that, very often, geometry/location information are specified within CAD drawings or text documents, etc. Referencing scheme may be added to relate an entity to the source that defines the location.

In the reference NEESgrid data model, Location is defined as an abstract class to represent a place where some objects, equipment or sensors are located or positioned. There are no slots defined for the Location class. A subclass of the base Location class is SpatialLocation, which is given in terms of spatial coordinates. The coordinate system can be either global or local, depending on if the relativeToLocation is specified. Detailed description of the SpatialLocation class is shown in the following table.

Slot Name	Type	Cardinality	Description
name	String	1	A word or set of alphanumeric characters by which any entity is identified or distinguished from others.
description	String	0:1	A simple description of the object.
coordX	Float	0:1	The X location given in terms of spatial coordinates.

coordY	Float	0:1	The Y location given in terms of spatial coordinates.
coordZ	Float	0:1	The Z location given in terms of spatial coordinates.
coordUom	:Unit	0:1	The unit of measurement for the defined coordinate system.
location Method	String	0:1	A description of the method by which the location reference point coordinates was obtained.
location Accuracy	String	0:1	An estimate of the accuracy of the location reference point. As this may be subjective, the description may be quantitative or qualitative.
relativeTo Location	:Location	0:1	The origin of the local coordinate system. If it is specified, the coordinate system will be local.

4.6.8 ApparatusLocation

The ApparatusLocation represents the particular location of an Apparatus. It is especially used for indicating the location of sensors. Detailed description of the ApparatusLocation class is shown in the following table.

Slot Name	Type	Cardinality	Description
hasApparatus	:Apparatus	1	The Apparatus (e.g. Sensor) that needs a location value.
theLocation	:Location	1	The physical location where the Apparatus resides.

5 Validation and Usability Test

The usability of the reference data model has been tested with legacy experimental data. At the time of the validation tests, Project Browser and data ingestion tools were under development and were not available. Therefore, Protégé [10] was employed as the interface to input experimental data and local file system was used as the storage medium. For illustration purpose, this report focuses on the data set obtained from a Mini-MOST experiment [21].

5.1 Mini-MOST Experiment

The main purpose of the Mini-MOST experiment is to show the capability of the various NEESgrid service components using a small-scale physical experimental setup [21]. The Mini-MOST experimental hardware, as implied by its name, is small in size and can be easily packed and shipped to experimental sites. The Mini-MOST experiment provides a platform for students and researchers to become familiar with the NEESgrid software and to gain first-hand experience in using the NEESgrid services. The Mini-MOST experiment can also be utilized for educational demonstration and software installation debugging. For the validation test of the reference data model, the data were generated from a particular Mini-MOST test on February 28, 2004, at the University of Illinois at Urbana-Champaign.

5.2 Inputting Experimental Data

Experimental data from the Mini-MOST experiment was ingested using Protégé [10] and saved as files in a local file system. Figure 16 shows loading an example project named miniMOST-1 into the system. Data are inputted using the slots (properties) as defined in the reference data model. If a slot is defined as primitive type, such as Integer, Real Number, Time, or String, etc., we can simply type in the value. If a slot is defined as Objects, then we can either choose a previously created object or create a new one. If a slot is defined as type “URI” (which would normally refer to a file), we can save the

particular file by entering the URI for the file location. Other types of objects, such as Task, EventGroups, Event, SensorSetup, InfrastructureSetup, Sensor, Specimen, and etc., can be created and input through an interface similar to the one shown in Figure 16. All the objects related to Mini-MOST experiment have been created and saved; the metadata and information about the data are saved as an OWL (Web Ontology Language) (<http://www.w3.org/2001/sw/WebOnt/>) file. Other experimental data, such as specimen photos and sensor readings, can be stored in a file on a web server with its URI saved in the OWL file.

5.3 Browsing Experimental Data

For validation purpose, we implemented a project viewer to retrieve the saved data and to view the data on a web browser according to the data model. The program is implemented using Java Servlet technology (<http://java.sun.com/products/servlet/>), and the parsing of the OWL file is handled by using Jena [19]. Figure 17 shows the front page of the project viewer with a list of saved projects. When we click on a particular project, say miniMOST-1, the details of the project will be shown on the browser, as illustrated in Figure 18.

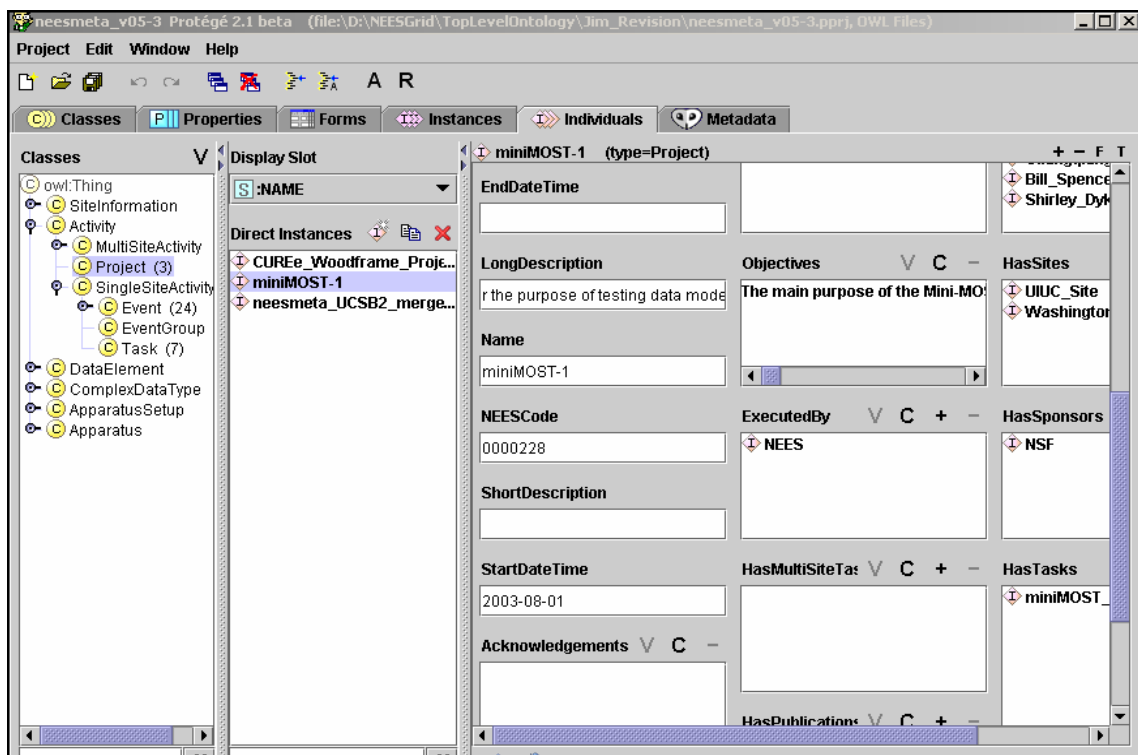


Figure 16 – Using Protégé to Input Mini-MOST Experiment Data

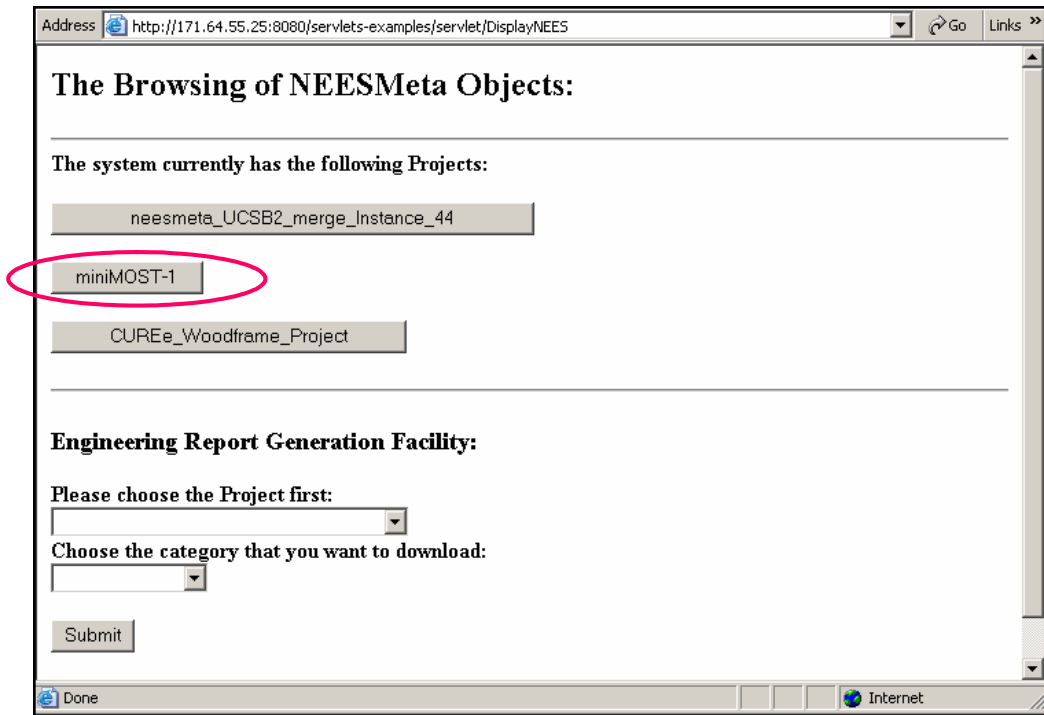


Figure 17 – The Front Page of the Project Viewer

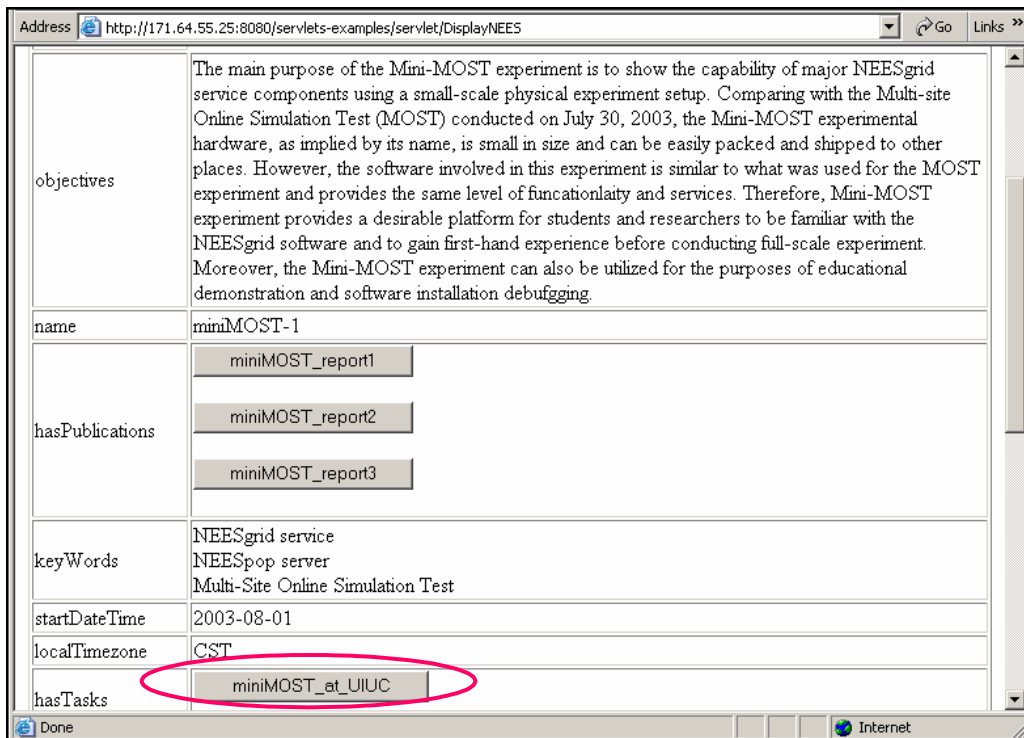
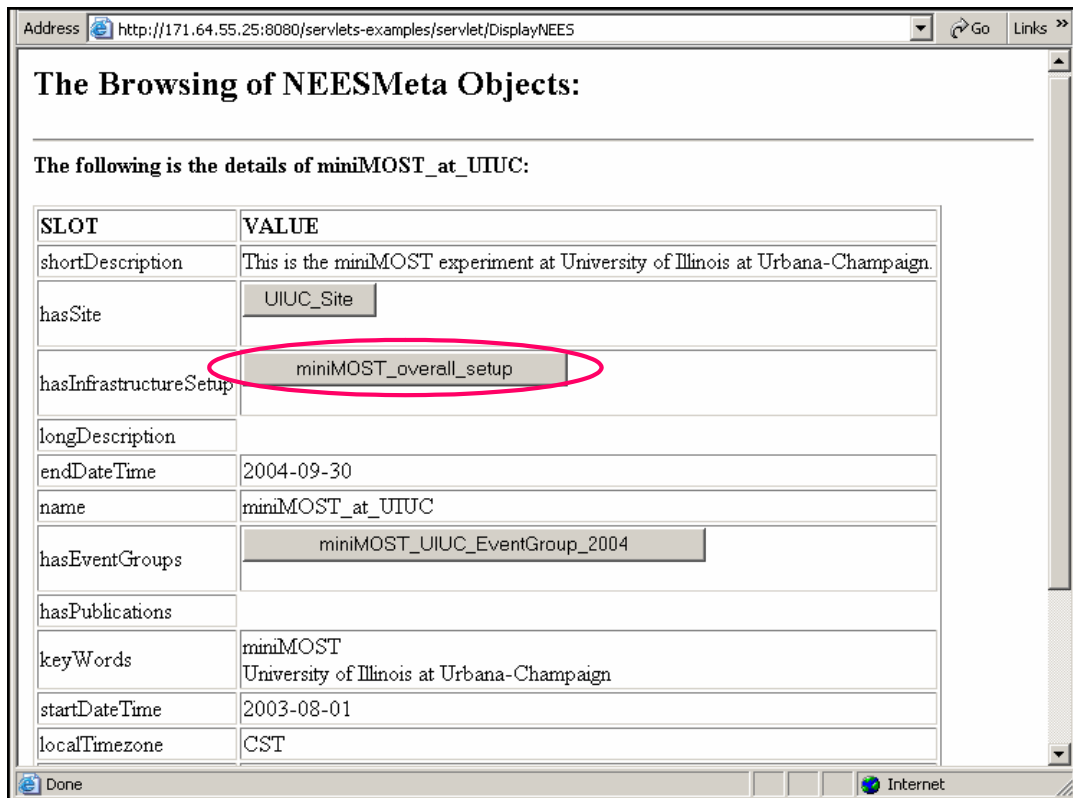


Figure 18 – Detailed Display of the Project MiniMOST-1

As defined in the reference data model, a Project is a collection (organized group) of Tasks designed to achieve specific goals and objectives. Following the model, we can navigate and access all the Tasks that belong to the Project. Figure 19 shows the details of a particular Task named miniMOST_at_UIUC. One property (or a slot) of a Task object is InfrastructureSetup, which models the assembly and arrangement of the PrimaryEquipment used for a specific Task. We can access the details of the InfrastructureSetup object by clicking on the highlighted button as shown in Figure 19.

Figure 20 presents the details of the InfrastructureSetup, which essentially is a collection of texts, documents (in the format of Word, PDF, Excel, etc.), figures and drawings stored as files. Files are saved in a web server and their URIs are saved as metadata. The files can be dynamically downloaded and shown on a web browser, as illustrated in Figure 21.

Each Task in a project may contain one or more EventGroups. The EventGroup object can be accessed by clicking on the highlighted button shown in Figure 22. The details of a particular EventGroup object named miniMOST_UIUC_EventGroup_2004 are presented in Figure 23.



SLOT	VALUE
shortDescription	This is the miniMOST experiment at University of Illinois at Urbana-Champaign.
hasSite	UIUC_Site
hasInfrastructureSetup	miniMOST_overall_setup
longDescription	
endDateTime	2004-09-30
name	miniMOST_at_UIUC
hasEventGroups	miniMOST_UIUC_EventGroup_2004
hasPublications	
keyWords	miniMOST University of Illinois at Urbana-Champaign
startDateTime	2003-08-01
localTimezone	CST

Figure 19 – Detailed Display of the Task miniMOST_at_UIUC

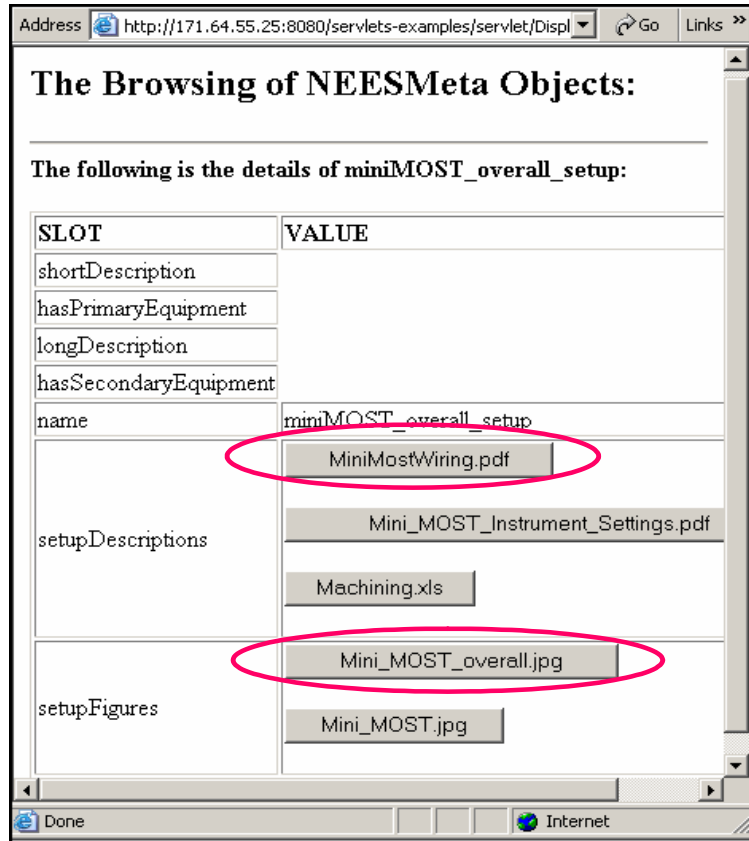
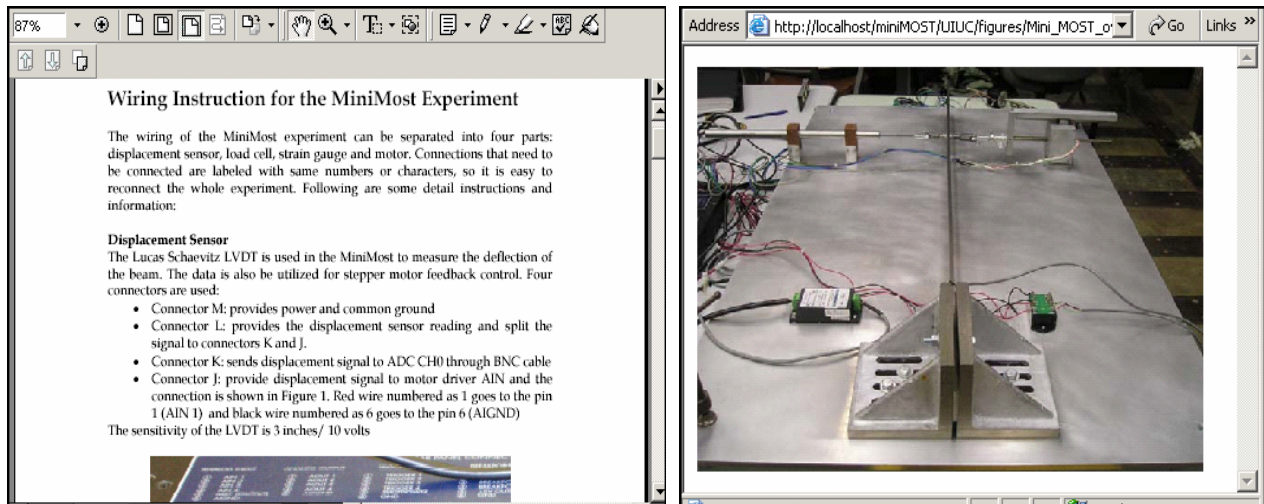


Figure 20 – Detailed Display of the InfrastructureSetup



(a) MiniMostWiring.pdf

(b) Mini_MOST_overall.jpg

Figure 21 – Access of Files Representing the InfrastructureSetup

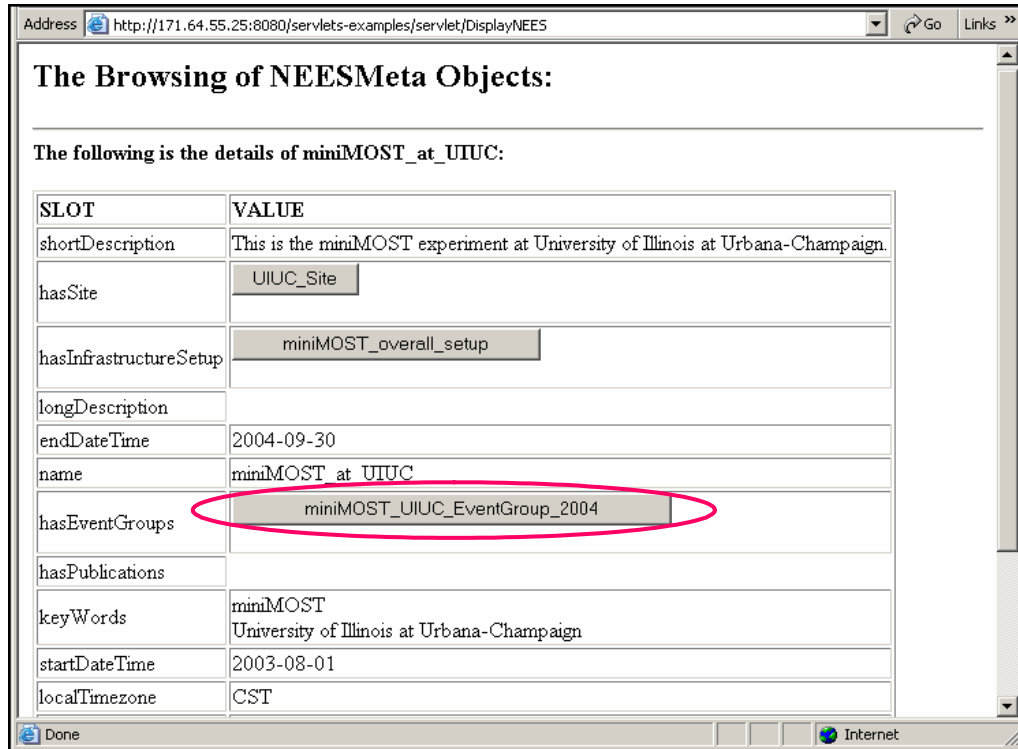


Figure 22 – Detailed Display of the Task miniMOST_at_UIUC

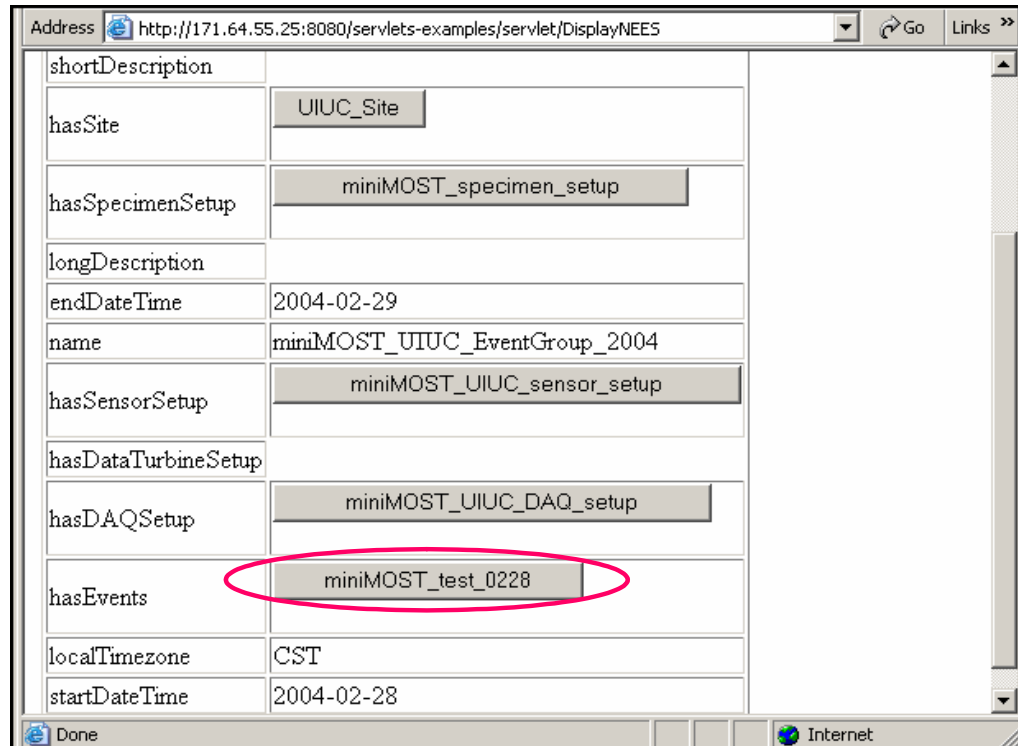


Figure 23 – Detailed Display of the EventGroup

An EventGroup is defined as a collection of Events, each of which can be accessed from the EventGroup object. The details of an example Event named miniMOST_test_0228 are shown in Figure 24. An Event, which is the atomic level of Activity, refers to each single run of an experiment or a simulation. Experimental results, such as SensorReading, can be accessed from an Event object, as shown in Figure 25.

SLOT	VALUE
shortDescription	An event with one-acurator miniMOST setup.
hasSite	UIUC_Site
testType	pseudo dynamic
hasWaveFormSetup	
longDescription	
endDateTime	2004-02-29T02:11:14.78099
name	miniMOST test 0228
hasOutputData	miniMOST_test_0228_results
startDateTime	2004-02-28T20:15:49.57800
loadTimezone	CST

Figure 24 – Detailed Display of the Event miniMOST_test_0228

```

Active channels: LVDT, StrainGage, LoadCell
Channel units: m, microstrain, N

Time      LVDT      StrainGage      LoadCell
2004-02-28T20:15:49.57800      -0.000072      -20.202637      0.737591
2004-02-28T20:15:58.03099      -0.000063      -17.211914      0.836224
2004-02-28T20:16:05.59299      -0.000053      -18.432617      0.840723
2004-02-28T20:16:13.01499      -0.000077      -17.395020      0.836664
2004-02-28T20:16:20.43699      -0.000060      -17.456055      0.734262
2004-02-28T20:16:28.12500      -0.000063      -17.456055      0.836754
2004-02-28T20:16:36.76499      -0.000060      -17.944336      0.735809
2004-02-28T20:16:44.31199      -0.000086      -20.080566      0.838325
2004-02-28T20:16:52.07800      -0.000079      -17.028809      0.838468
2004-02-28T20:16:59.50000      -0.000098      -18.920898      0.933884
2004-02-28T20:17:06.98399      -0.000107      -16.296387      0.928718
2004-02-28T20:17:14.51499      -0.000121      -18.920898      1.034670
2004-02-28T20:17:21.90599      -0.000126      -15.747070      1.031043
2004-02-28T20:17:29.31199      -0.000147      -16.540527      1.036738
2004-02-28T20:17:36.75000      -0.000144      -17.578125      1.031765
2004-02-28T20:17:44.28099      -0.000149      -15.136719      1.035167
2004-02-28T20:17:52.71799      -0.000135      -15.319824      1.038448
2004-02-28T20:18:00.13999      -0.000147      -17.333984      1.035462
2004-02-28T20:18:07.56199      -0.000102      -17.150879      0.932433
2004-02-28T20:18:15.07800      -0.000058      -19.042969      0.836102
    
```

Figure 25 – Access of SensorReading for the Event miniMOST_test_0228

The EventGroup object also contains the objects of SpecimenSetup, SensorSetup, and DAQSetup. Figure 26 shows the details of the SensorSetup object, which belongs to the EventGroup named miniMOST_UIUC_EventGroup_2004. Again, the setup is described in texts, documents, drawings and picture files. Each file can be accessed by simply following the URI for the file. For example, Figure 27 shows a photo for the setup of a LVDT sensor.

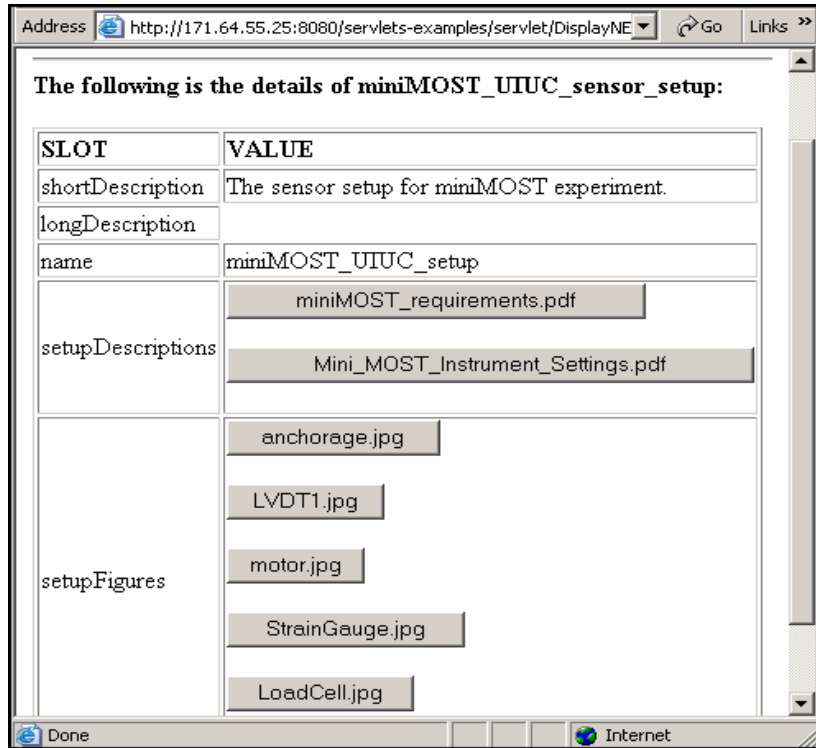


Figure 26 – Detailed Display of the SensorSetup

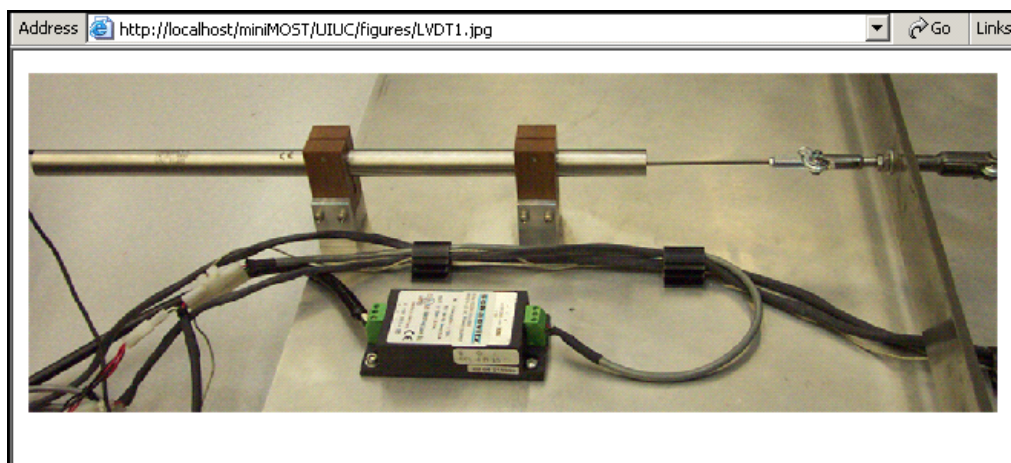


Figure 27 – Access of Photo for the LVDT

6 Summary and Discussions

In this report, details of a reference NEESgrid data model have been presented. The intention of this document is to give a description of the data model and to solicit feedback and comments from the NEES community. Although the reference data model described in this report focuses on shake table experiments, many of the features can be applied or extended to centrifuge, field tests, tsunami, pseudo-dynamic and other types of experiments. Six base classes and their subclasses are presented, and the relationships among these classes are defined. These classes represent the essential elements to support the end-to-end solution of NEESgrid data efforts. We believe the proposed reference data model is flexible and extendible: (1) new classes can easily be introduced; (2) the slots of a particular class can be added, deleted, or modified; and (3) the relationships among the classes can be altered. Other models, such as specimen model, unit model, geometry/location model, and Site model, can be appended to (or even replace certain parts of) the reference data model.

To validate the reference data model, we have populated the model with the mini-MOST experimental data provided by UIUC. This validation process helps evaluate the completeness, flexibility and usability of the data model. The usability test has demonstrated that the data model is sufficiently comprehensive to save and organize all the mini-MOST data. In addition, as the experimental data are organized according to the data model, browsing and accessing them are fairly intuitive and straightforward. Efforts will continue to validate, evaluate and refine the reference data model using other experimental projects and data. In addition, a general Project Browser for the ingestion and browsing of NEESgrid data is currently under development by the NEES System Integrator team.

We would like to emphasize that data model development is a community effort. Suggestions and feedback from the NEES community and stakeholders are important in the development process. The reference data model described in this document is based on version 1.0, which has been released for the review of the NEES community at the beginning of July 2004. Data model development is an iterative and evolving process, and the reference data model will continue to be tested, validated, modified and revised, even beyond the current development effort. We look forward to receiving and to incorporating any valuable suggestions from the NEES community.

The development of NEESgrid data models, together with the current efforts of developing data ingestion tools, the data repository, and tools that directly support experimental activities, all serve as initial steps towards data sharing, archival and curation. Data curation implies well-planned active management of information and involves the production, conservation, preservation and access of the data [18]. The active management of data must ensure that the people to whom the data is relevant can find the data. Furthermore, data curation needs to ensure supports of data/information reuse and facilitate generation of new information and knowledge from the saved data. Continuing developments in data curation effort are recommended.

Acknowledgements

This report is drafted by the authors as part of the NEES System Integration effort, WBS No. 2.4 Data and Metadata Management. The authors would like to acknowledge the active collaboration and contributions of the NEESgrid's Data/Metadata task committee members (in alphabetical order by their first name):

Andrei Reinhorn	State University of New York, Buffalo
Bill Spencer	University of Urbana-Champaign
Chuck Severance	University of Michigan
Gokhan Pekcan	University of Nevada, Reno
Hank Ratzesberger	University of California, Santa Barbara
Jean-Pierre Bardet	University of Southern California

Jennifer Swift	University of Southern California
Jim Eng	University of Michigan
Jun Peng	Stanford University
Ken Ferschweiler	Northwest Alliance for Computational Science and Engineering
Kincho H. Law	Stanford University
Lelli Van Den Einde	University of California, San Diego

The authors would first and foremost acknowledge Professor Gokhan Pekcan of University of Nevada, Reno for his help in developing the reference data model, and Professor Bill Spencer of University of Illinois, Urbana-Champaign and Dr. Cristina Beldica of National Center for Supercomputing Applications (NCSA) for their encouragements and supports.

The authors would like to thank Joe Futrelle of NCSA, Chuck Severance and Jim Eng of University of Michigan for their time and discussions related to NEESgrid developments. The authors would also like to thank Professor Gokhan Peckan and Dr. Patrick Laplace of University of Nevada, Reno; Professors Steve Mahin, Bozidar Stojadinovic, Greg Fenves, and Dr. Frank McKenna of University of California, Berkeley; Professor Andrei Reihorn of SUNY-Buffalo; and Professors Jerome Hajjar and Catherine French of University of Minnesota for their time to discuss the data and metadata issues related to earthquake engineering experiments and simulations. Any opinions, findings, and conclusions or recommendations expressed in this material are, however, those of the authors and do not necessarily reflect the views of others and the National Science Foundation.

References

- [1] NEESgrid Team. *Multi-site Online Simulation Test (MOST)*, 2003. (<http://www.neesgrid.org/most/index.html>)
- [2] Oregon State University and Network Alliance for Computational Science and Engineering. *NEES Database and Metadata Structure, Version 1.3*, white paper, Network for Earthquake Engineering Simulation, 2003.
- [3] J. Arlow and I. Neustadt. *UML and the Unified Process: Practical Object-Oriented Analysis and Design*, Addison-Wesley Pub Co., Boston, MA, 2001.
- [4] R. Benjamins, D. Fensel, and A. G. Perez. "Knowledge Management through Ontologies," *Proceedings of the 2nd International Conference on Practical Aspects of Knowledge Management*, Basel, Switzerland, 1998.
- [5] M. Botts. *Sensor Model Language (SensorML) for In-situ and Remote Sensors*, OpenGIS Interoperability Program Report, OGC 02-026, Open GIS Consortium Inc, 2002. (http://vast.uah.edu/SensorML/OGC-02-026_SensorML_0.07.doc)
- [6] M. L. Brodie. "Association: A Database Abstraction for Semantic Modeling," *Proceedings of 2nd International Entity-Relationship Conference*, Washington, DC, 1981.
- [7] C. M. Eastman. *Building Product Models: Computer Environments, Supporting Design and Construction*, CRC Press, 1999.
- [8] G. L. Fenves and F. McKenna. *Data Model for Simulation*, Technical Report NEESgrid-2004-46, 2004. (http://www.neesgrid.org/documents/TR_2004_46.pdf)
- [9] D. Fischer, A. Filiatrault, B. Folz, C.-M. Uang, and F. Seible. *CUREE-Caltech Woodframe Project: Shake Table Tests of a Two-Story Woodframe House*, Consortium of Universities for Research in Earthquake Engineering, 2001.

- [10] J. Gennari, M. A. Musen, R. W. Fergerson, W. E. Grosso, M. Crubézy, H. Eriksson, N. F. Noy, and S. W. Tu. *The Evolution of Protégé: An Environment for Knowledge-Based Systems Development*, Stanford Medical Informatics, Stanford University, 2002. (http://smi.stanford.edu/pubs/SMI_Abstracts/SMI-2002-0943.html)
- [11] IAI. *Industry Foundation Classes, Specification Volumes 1-4*, International Alliance for Interoperability, Washington, DC, 1997.
- [12] ISO. *Product Data Representation and Exchange, Part 1: Overview and Fundamental Principles*, No. 10303-1, International Organization for Standardization, 1994.
- [13] B. L. Kutter, D. W. Wilson, C. Pancake, and S. Haerer. *Introduction to the Site Specifications Database (SSDB)*, Network for Earthquake Engineering Simulation, 2004. (<http://nees.orst.edu/IT/site.specs.db/cohorts/Introduction.pdf>)
- [14] K. H. Law and M. K. Jouaneh. "Data Modeling for Building Design," *Proceedings of the 4th Computing Conference in Civil Engineering*, Boston, MA, 1986.
- [15] K. H. Law, M. K. Jouaneh, and D. L. Spooner. "Abstraction Database Concept for Engineering Modeling," *Engineering with Computers*, 2:79-94, 1987.
- [16] K. H. Law. "Conceptual Database Design for Engineering Modeling," *Proceedings of The ASME International Computers in Engineering Conference and Exhibition, Managing Engineering Data: Emerging Issues*, San Francisco, CA, 1988.
- [17] K. H. Law, G. Wiederhold, N. Siambela, W. Sujansky, D. Zingmond, H. Singh, and T. Barsalou. "Architecture for Managing Design Objects in a Sharable Relational Framework," *International Journal of Systems Automation: Research and Applications (SARA)*, 1:47-65, 1991.
- [18] K. H. Law. *Summary Report on NEESgrid's Data Curation Summit*, Technical Report NEESgrid-2004-43, 2004. (http://www.neesgrid.org/documents/TR_2004_43.pdf)
- [19] B. McBride, D. Boothby, and C. Dollin. *An Introduction to RDF and the Jena RDF API*, 2004. (http://jena.sourceforge.net/tutorial/RDF_API/index.html)
- [20] D. L. McGuinness and F. v. Harmelen. *OWL Web Ontology Language Overview*, W3C Recommendation 10 February 2004, <http://www.w3.org/TR/owl-features/>, 2004.
- [21] N. Nakata, G. Yang, and B. F. Spencer. *System Requirements for Mini-MOST Experiment*, NEESgrid Technical Report, 2004. (http://www.neesgrid.org/mini-most/Mini_MOST_requirements_revised3.pdf)
- [22] N. F. Noy and D. L. McGuinness. *Ontology Development 101: A Guide to Creating Your First Ontology*, Stanford University, Stanford, CA, 2002. (http://protege.stanford.edu/publications/ontology_development/ontology101.html)
- [23] J. Peng and K. H. Law. *A Brief Review of Data Models for NEESgrid*, Technical Report NEESgrid-2004-01, 2004. (http://www.neesgrid.org/documents/TR_2004_01.pdf)
- [24] J. Peng and K. H. Law. *Validity and Usability of the NEESgrid Reference Data Model*, Technical Report NEESgrid-2004-44, 2004. (http://www.neesgrid.org/documents/TR_2004_44.pdf)
- [25] J. Peng and K. H. Law. "A Reference Data Model for NEESgrid Shake Table Experiments," *Proceedings of the International Symposium on Earthquake Engineering in the Past and Future Fifty Years*, Harbin, China, 2004.

- [26] J. Peng, K. H. Law, and G. Pekcan. *Reference NEESgrid Data Model for Shake Table Experiment*, NEESgrid Technical Report, 2004.
- [27] J. R. Rumbaugh, M. R. Blaha, W. Lorenzen, F. Eddy, and W. Premerlani. *Object-Oriented Modeling and Design*, Prentice Hall, 1990.
- [28] J. M. Smith and D. C. P. Smith. "Database Abstractions: Aggregation and Generation," *ACM Transaction on Database Systems*, 2(2):105-133, 1977.
- [29] J. Sowa. *Conceptual Structures: Information Processing in Minds and Machines*, Addison-Wesley, Reading, MA, 1984.
- [30] J. Swift, J.-P. Bardet, and G. Pekcan. *Reference NEESgrid Data Model for Centrifuge Experiments*, NEESgrid Technical Report, 2004.